Sparse Methods in Image Understanding and Computer Vision

by

Jayaraman Jayaraman Thiagarajan

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved March 2013 by the
Graduate Supervisory Committee:

Andreas Spanias, Chair
David Frakes
Cihan Tepedelenlioglu
Pavan Turaga

ARIZONA STATE UNIVERSITY

May 2013

ABSTRACT

Image understanding has been playing an increasingly crucial role in vision applications. Sparse models form an important component in image understanding, since the statistics of natural images reveal the presence of sparse structure. By representing data as a sparse linear combination of atoms from a "dictionary" matrix, these methods lead to parsimonious models, in addition to being efficient for large scale learning.

This dissertation focuses on understanding different aspects of sparse learning, and enhancing sparse methods by incorporating tools from machine learning. With the growing need to adapt models for large scale data, it is important to design dictionaries that can generalize to the entire probability space of data using a finite training set. By exploring the relation between dictionary learning and one-dimensional subspace clustering, a multilevel dictionary learning algorithm is developed, and it is shown to outperform conventional sparse models in compressed recovery, and image denoising. Theoretical aspects of learning such as algorithmic stability and generalization are considered, and ensemble methods are incorporated for effective large scale learning. In addition to building strategies for efficiently implementing one-dimensional subspace clustering, a discriminative clustering approach is designed to estimate the unknown mixing process in blind source separation.

By exploiting the non-linear relation between the image descriptors, and allowing the use of multiple features, sparse methods can be made more effective in recognition problems. The idea of multiple kernel sparse representations is developed, and algorithms for learning dictionaries in the feature space are presented. Using object recognition experiments on standard datasets it is shown that the proposed approaches outperform other sparse coding-based recognition frameworks. Furthermore, a segmentation technique based on multiple kernel sparse representations is developed, and successfully applied for automated brain tumor identification. Using

sparse codes to define the relation between data samples can lead to a more robust graph embedding for unsupervised clustering. By performing discriminative embedding using sparse coding-based graphs, an algorithm for measuring the glomerular number in kidney MRI images is developed. Finally, approaches to build dictionaries for local sparse coding of image descriptors are presented, and applied to object recognition and image retrieval.

To

my dearest family and friends for

their unconditional love ...

ACKNOWLEDGEMENTS

I would like to express the deepest appreciation to my PhD advisor, Professor Andreas Spanias, for his unending support and trust. Over the years, he has continually enriched my graduate studies, and nurtured my passion towards scientific research. My fortuitous collaborations and a diverse research experience would not have been possible, if not for the immense support from Prof. Spanias.

I would also extend my gratitude to my committee members, Dr. Cihan Tepedelenlioglu, Dr. Pavan Turaga, and Dr. David Frakes, for their constant support, and their insightful feedback on my research. My collaborations with Dr. Turaga and Dr. Frakes have been very exciting, and both of them have been sources of inspiration for my research pursuits. Special thanks to Professor Antonia Papandreou-Suppappola for her help and support. I would like to thank all professors who have taught me courses at ASU. I am grateful to Dr. Darin Taverna, my mentor at Translational Genomics Research Institute, for his guidance and continual support. I sincerely thank Professor P.P. Vaidyanathan (Caltech) for the "inspiring" discussion with him at the 2011 Asilomar conference. Special thanks to Darleen, Cynthia, Merri, Donna, Ginger, and Jenna for all their help over the years. I thank all my lab colleagues at ASU for their camaraderie, support and more than anything, for inspiring me with their passion towards research.

The seeds for my interest in research were planted as an undergraduate student in India, and I owe that largely to Prof. Vetrivel and his adventurous spirit towards science and engineering. I would also like to thank Mr. Madhusudan Shekar, my project manager at IBM (India), for his support and enthusiasm.

My heartfelt gratitude to my family, for their unrelenting support and unconditional love towards me. If not for their determination, I would not have been to able to pursue graduate studies in the first place. I cannot thank them enough for their patience, and enthusiasm towards any of my tiniest accomplishments.

My dearest friends deserve my gratitude for continually making me a better person with their care and support. I feel fortunate to have had my friends Karthi, Harini, and Vivek, who have made me an integral part of their lives. Special thanks to Deepta, Balamurugan, Angel, Deepan, Navaneethakrishnan, Dilip, Harish, Omar, Prathap, Deborah, Mahesh, Suhas, and Prasanna for their unconditional support.

Finally, I thank every single person in my life who has inspired me to follow my heart, beyond the odds.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Natural images are pervasive entities that have been studied over the past five decades. The term *natural images* encompasses a general class that includes scenes and objects present in nature as well as man-made entities. Natural images have interested scientists from a wide range of fields due to the fact that their local regions possess substantial similarities, in spite of their rich variability when looked as a whole. Some examples of natural images are illustrated in Figure 1.1. Several proposed models for representing natural images have tried to mimic the processing of the human visual system. In general, image representation is concerned with low-level vision. Therefore, using local regions or *patches* from images and exploiting the local similarities in order to build features has resulted in efficient representation. The representative features that are extracted from local image regions are referred to as *low-level features*. It is imperative that we understand and incorporate our knowledge of the human visual processing as well as the local image statistics when building such systems for representation of images.

1.1 Natural Image Statistics

A digital image denoted using the symbol I is represented using pixel values between 0 and 255, for grayscale. Image statistics refer to the various statistical information of the raw pixel values, or their transformed versions. By incorporating natural image statistics as prior information in a Bayesian inference scheme, one can hope to learn features that are adapted to the image data rather than trying to use classical transform methods that may not be adapted well to the data.

Since analyzing raw pixel statistics hardly produces any useful information, analysis of natural image statistics has been performed in Fourier [7] and Wavelet domains [8]. It is observed that the power spectrum changes as $1/f^2$, where f is

Figure 1.1: Examples of natural images obtained from the Berkeley segmentation dataset (BSDS) [1].

the frequency, and this has been one of the earliest proofs of redundancy in natural images. The marginal statistics of wavelet coefficients in a subband show that the distributions are more peaky and heavily tailed when compared to a Gaussian distribution. Furthermore, the joint statistics indicate strong dependence between the coefficients in different subbands. The marginal statistics of wavelet coefficients for a subband shown in Figure 1.2 demonstrates this behavior. The coefficient statistics obtained using log-Gabor filters on natural images also reveal a peaked distribution. Figure 1.3 shows the output distribution at scale 1 and orientation 1 for the *Barbara* image.

Though the statistics of full natural images convey useful information, it is also helpful to consider the distribution of natural image patches in space. It has been shown that high contrast image patches lie in clusters and near low-dimensional manifolds [9, 10]. Patches of size 3×3 were chosen from natural images and preprocessed in the logarithm scale to remove the mean and normalize the contrast values. They were then projected using the discrete cosine transform (DCT) basis, and normalized to unit norm, which makes the processed data lie in a $7-$sphere in $\mathbb{R}^8$. By computing the fraction of the data points that near the dense sampling of the surface of the sphere, the empirical distribution of the data can be determined. It has been observed that a majority of the data points are concentrated in a few high density regions of the sphere. The sample points corresponding to the high density regions are

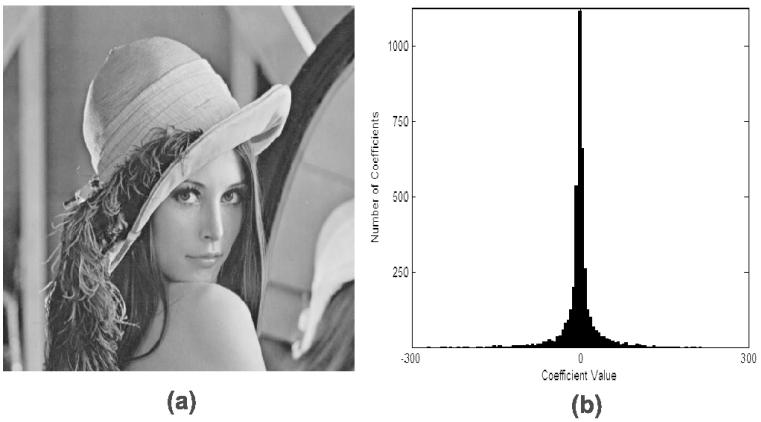


Figure 1.2: (a) *Lena* image (b) histogram of the wavelet coefficients in scale 3.

similar to blurred step edges for natural image patches. Further topological analysis of natural image patches has been performed in [10], which suggests that features that can efficiently represent a large portion of natural images can be extracted by computing topological equivalents to the space of natural image patches.

1.2 Sparseness in Biological Vision

Most of our cognitive functions and perceptual processes are carried out in the neocortex, which is the largest part of the human brain. The primary visual cortex, also referred to as V1, is the part of the neocortex that receives visual input from the retina. The nobel prize winning discoveries of Hubel and Weisel showed that the primary visual cortex consists of cells responsive to simple and complex features in the input. V1 has receptive fields that are characterized as being spatially localized, oriented and bandpass. In other words, they are selective to structure of the visual input at different spatial scales. One approach to understanding the response properties of visual neurons has been to consider their relationship to the statistical structure of natural images in terms of efficient coding.

A generative model that constructs random noise intensities typically results in images with equal probability. However, the statistics of natural image patches have been shown to contain a variety of regularities. Hence, the probability of generating

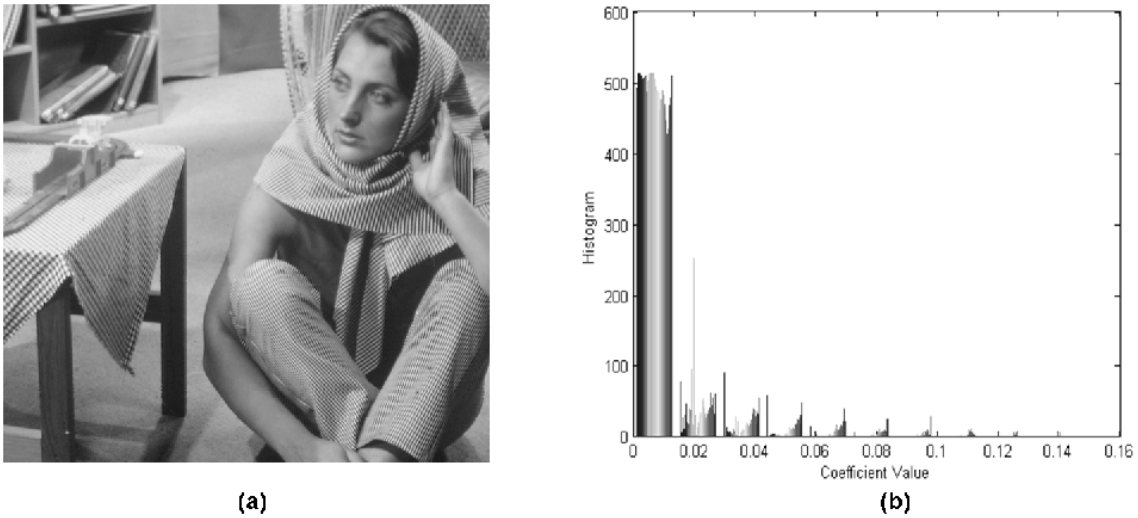


Figure 1.3: (a) The *Barbara* image, (b) log-Gabor coefficient statistics at scale 1 and orientation 1 for the image.

a natural image using the random noise model is extremely low. In other words, the amount of redundancy in natural images allows the design of a visual system that can efficiently represent the natural scenes. Extending this argument, understanding the behavior of the visual system in exploiting redundancy, will enable to us to build a plausible explanation for the coding behavior of visual neurons [11].

The optimality of the visual coding process can be addressed with respect to different metrics of efficiency. One of the most commonly used metrics is the *representation efficiency*. Several approaches have been developed to explore the properties of neurons involved in image representation. In [12], it was first reported that the neurons found in the V1 showed similarities to Gabor filters, and hence different computational models were developed based on this relation [13, 14]. Furthermore, the $1/f^k$ fall off observed in the Fourier spectra of natural images demonstrated the redundancy in the images. The $1/f$ structure arises because of the pairwise correlations in the data, and the observation that natural images are approximately scale invariant. The pairwise correlations account for about 40% of the total redundancy in natural scenes. Any representation with significant correlations implies that most

of the signal lies in a subspace within the larger space of possible representations [15] and hence data can be represented with much reduced dimensionality. Though pairwise correlations have been an important form of redundancy, studies have shown that there exists a number of other forms of redundancy. Two images with similar $1/f$ spectra can be described in terms of differences in their sparse structure.

For example, for a noise image with $1/f$ structure, all linear representations will result in a Gaussian response distribution. However, with natural images, the histogram of the responses will be non-Gaussian for a suitable choice of linear filters (Figure 1.3). A visual system that generates such response distributions, with high kurtosis (fourth statistical moment), can produce a representation with visual neurons that are maximally independent. In other words, codes with maximal independence will activate neurons with maximal unique information. This is the principal idea behind sparse coding algorithms. With respect to biological vision, *sparsity* implies that a small proportion of the neurons are active, and the rest of the population is inactive with high probability. Olshausen and Field showed that by designing a neural network that attempts to find sparse linear codes for natural scenes, we can obtain a family of localized, oriented and bandpass basis functions similar to those found in the primary visual cortex. This evidenced that at the level of V1, visual system representations can efficiently match the sparseness of natural scenes. However, it must be emphasized that the sparse outputs of these models result from the sparse structure of the data. For example, similar arguments cannot be made for white noise images. Further studies have shown that the visual neurons produce sparse responses in higher stages of cortical processing such as inferotemporal cortex, in addition to the primary visual cortex. However, there is no biological evidence that shows that the sparse responses imply efficient representation of the environment by the neurons.

Measuring the efficiency of neural systems is very complicated compared to engineered visual systems. In addition to characterizing the visual representations, there is a need to understand its dependency on efficient learning and development. A learning algorithm must address a number of problems pertinent to generalization. For example, invariance is an important property that the learning algorithm must possess. In this case, efficient learning is determined by its ability to balance between the selection of suitable neurons and achieving invariance across examples with features that vary. In other words, it is critical for the algorithm to generalize to multiple instances of an object in the images. In one end, this problem can be addressed by building a neuron for every object. Simpler tasks such as distinguishing between different faces can be efficiently performed using this approach. However, the number of object detectors in such a system would be exorbitantly high. On the other end, sparse codes with more neurons are necessary for visually challenging tasks such as object categorization. Hence, the general task of object recognition requires different strategies with varying degrees of sparseness.

Another important property of the representations in biological vision is that they are highly overcomplete. As expected, these overcomplete codes involve significant redundancy. Overcompleteness has been suggested as an efficient way to model the redundancy in images [16]. Furthermore, overcompleteness can result in highly sparse representations, when compared to using complete codes, and this property is very crucial for generalization during learning. Though a complete theory to describe the human visual processing still eludes the neuro-scientists, it has been found that considering the different metrics of efficiency together is crucial while addressing this problem.

1.3 The Generative Model for Sparse Coding

The linear generative model for sparse representation of a data sample $\mathbf{x} \in \mathbb{R}^M$ is given by,

$$\mathbf{x} = \mathbf{D}\mathbf{a} \tag{1.1}$$

where $\mathbf{D} \in \mathbb{R}^{M \times K}$ is the set of K elementary features and $\mathbf{a} \in \mathbb{R}^K$ is the coefficient vector. If we assume that the coefficient vector is sparse and has statistically independent components, the elements of the dictionary $\mathbf{D}$ can be inferred from the generative model using appropriate constraints 8.9. The reason for assuming a sparse prior on the elements of $\mathbf{a}$ is the intuition that natural signals and images allow for their efficient representation as a sparse linear combination of patterns such as edges, lines and other elementary features [17]. In addition, the experiments given in Section 1.1 that analyze the statistics of wavelet and log-Gabor coefficients are an evidence for sparseness of the coefficient vector. Let us assume that all the T patches extracted from the image I are given by the matrix $\mathbf{X} \in \mathbb{R}^{M \times T}$, and the coefficient vectors for all the T patches are given by the matrix $\mathbf{A} \in \mathbb{R}^{K \times T}$. The likelihood for the image I, which is represented by $\mathbf{X}$ is given by

$$\log p(\mathbf{X}|\mathbf{D}) = \int p(\mathbf{X}|\mathbf{D}, \mathbf{A}) p(\mathbf{A}) d\mathbf{A}. \tag{1.2}$$

$p(\mathbf{X}|\mathbf{D}, \mathbf{A})$ can be computed using the linear generative model assumed in (8.9) and $p(\mathbf{A})$ is the prior that enforces sparsity constraints on the entries of the coefficient vectors. The dictionary now can be inferred as

$$\hat{\mathbf{D}} = \underset{\mathbf{D}}{\operatorname{argmax}} \log p(\mathbf{X}|\mathbf{D}). \tag{1.3}$$

The strategy proposed by Olshausen and Field [18] to determine the dictionary $\mathbf{D}$ is to use the generative model (8.9) and an approximation of (1.3) along with the constraint that the columns of $\mathbf{D}$ are of unit ℓ_2 norm. The dictionary $\mathbf{D}$ here is over-complete (i.e.) the number of dictionary elements is greater than the dimensionality

of the image patch, $K > M$. This leads to infinite solutions for the coefficent vector in (8.9) and hence the assumption on sparsity of $\mathbf{a}$ can be used to choose the most appropriate dictionary elements that represent $\mathbf{x}$. Our strategy for computing sparse representations and learning dictionaries will also be based on this generative model. Though the generative model assumes that the distribution of coefficients are independent, there will still be statistical dependence between the inferred coefficients. This is because the elementary features corresponding to the non-zero coefficients occlude partially each other.

It should be noted that using a model such as (8.9) along with sparsity constraints on $\mathbf{a}$ deviates from the linear representation framework. Assume that we have a representation that is $S-$sparse, (i.e.), only S out of K coefficients are non-zero at any time in $\mathbf{a}$. There are totally $\binom{K}{S}$ such combinations possible, each representing an $S-$dimensional subspace assuming that each set of S chosen dictionary elements are independent. This model leads to a union of $S-$dimensional subspaces where the signal to be represented can possibly reside. It is clearly not a linear model as the sum of signals lying in two $S-$dimensional subspaces can possibly lie in a $2S-$dimensional subspace.

1.4 Sparse Models for Image Reconstruction

Images can be modeled using sparse representations on predefined dictionaries as well as dictionaries learned from the image data. Widely used predefined dictionaries include the discrete cosine transform (DCT), wavelet, wedgelet, and curvelet dictionaries. Since a single predefined dictionary cannot completely represent the patterns in an image, using a combination of predefined dictionaries is helpful in some applications.

1.4.1 Dictionary Design

DCT dictionaries were one of the earliest dictionaries for patch-based image representation and still used as initial dictionaries in a variety of dictionary learning tasks. Overcomplete DCT dictionaries can be easily constructed and have been used with sparse representations, performing substantially better than orthonormal DCT dictionaries. The set of predefined dictionaries that have probably found the widest applications in multiscale image representation are the wavelet dictionaries. Wavelet coefficients have strong inter- and intra-scale dependencies and modeling them can be helpful in applications that exploit this additional redundancy. Many successful image coders consider either implicit or explicit statistical dependencies between the wavelet coefficients. For example, the JPEG-2000 standard [19] considers the neighboring coefficients in the adjacent subbands of the same scale jointly while coding.

Using sparsity models with learned dictionaries have been very successful. The simplest model that assumes the coefficients are independent gives rise to the K-SVD learning algorithm [20]. If the sparsity patterns appear in groups or blocks, block based sparse representations can be performed [21]. Additional structure can be imposed on group sparsity and each sparse pattern can be penalized accordingly. This gives rise to the structured sparsity frameworks and they are an area of ongoing research [22].

1.4.2 Data Clustering

The problem of dictionary optimization is a generalization of data clustering. In particular, the K-means and K-lines clustering algorithms are special cases of the general joint optimization problem of sparse coding and dictionary learning. K-means clustering seeks to cluster the data samples by minimizing the sum of intracluster distances across all clusters and K-lines clustering performs a least squares fit of K 1-D subspaces, referred to as hyperlines, to the training data [23]. In the K-

means clustering problem, it is assumed that each coefficient vector is 1-sparse (i.e.) has exactly one non-zero coefficient and the coefficient value is 1. In the K-lines clustering problem, 1 sparsity is still assumed for the coefficient vector, but the value of the non-zero coefficient itself is unconstrained.

In addition to computing the clustering accuracy, the behavior of clustering algorithms can be understood by analyzing the algorithmic stability. The general idea behind stability of a clustering algorithm is that the algorithm should produce clusters that are not too different when different i.i.d. training sets from the same probability space are used for training [24–26]. In [24], the authors show that a clustering algorithm is stable if there is a unique minimizer to the objective function. This notion is used in [25] to characterize the stability of K-means clustering algorithm. It is shown in [26] that when there is no unique global minimizer to the objective function, K-means is stable with respect to a change in $o(\sqrt{T})$ samples between two i.i.d. training sets of T samples each, where $T \to \infty$.

1.5 Sparse Models for Recognition

In addition to constraining overcomplete linear representations, sparse coding can be interpreted as learning the underlying data distribution with a sparsity prior. Note that, Deep Belief Networks (DBNs) have also been used to effectively infer the data distribution and extract features in an unsupervised fashion [27]. By imposing sparsity constraints on DBNs for natural images will result in features that closely resemble their biological counterparts [28]. The relevance of sparsity in machine learning systems has been studied extensively over the last decade. Sparse features have been known to be more likely separable in high-dimensional spaces [29], and hence helpful in classification tasks. However, empirical studies designed to evaluate the importance of sparsity in feature extraction for image classification argue that no performance improvement is gained by imposing sparsity, unless this sparsity

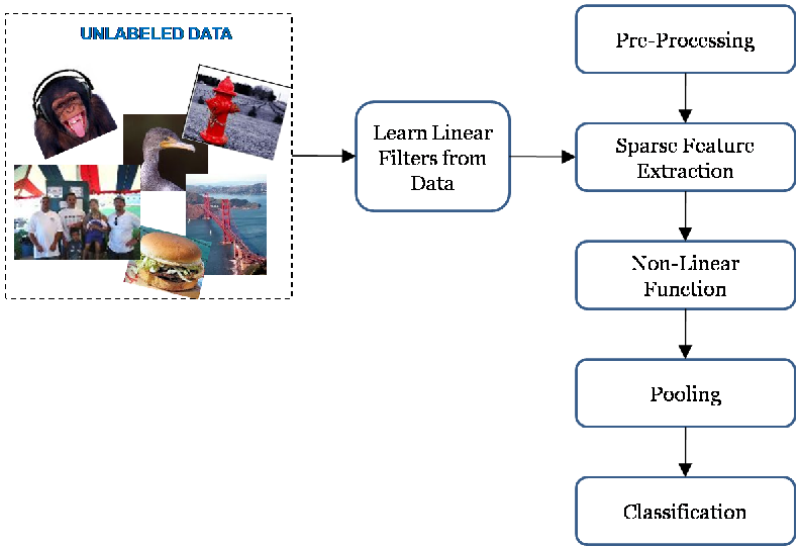


Figure 1.4: A biologically inspired standard object recognition architecture based on sparse representations.

is tailored for discrimination [30]. Let us consider a standard object recognition architecture based on sparse features, similar to the ones developed in [31,32]. Given a known set of filters (dictionary atoms), sparse features are extracted by coding local regions in an image. Note that, if sparsity does not result in an improved performance, the computationally intensive sparse coding process can be replaced by a simple convolution of the patches with the linear filters.

The image filters can be learned using dictionary learning procedures, or can be pre-determined based on knowledge about the data. The architecture illustrated in 1.4 has a pre-processing step where operations such as conversion of color images to grayscale, whitening, and dimensionality reduction can be carried out. Sparse feature extraction is followed by the application a non-linear function, such as taking the absolute values of the coefficients, and pooling. Note that, this procedure is common in biologically inspired multi-layer architectures for recognition. Additional operations such as downsampling can also be carried out. Since the dimension of the descriptors is too high for practical applications, downsampling is very crucial

for subsequent operations. Some of the commonly used pooling functions include Gaussian pooling, average pooling, and maximum value pooling (*max-pooling*) in the neighborhood. The final step in the architecture is to learn a classifier that discriminates the pooled features belonging to the different classes.

1.6 Problem Statement

This dissertation focuses on understanding different aspects of sparse coding and dictionary learning for effective image modeling. Furthermore, approaches that incorporate machine learning principles to augment sparse methods, for use in supervised and unsupervised learning problems, are developed.

1.6.1 Analysis and Implementation of 1-D Subspace Clustering

In the paradigm of sparse representations, analysis of 1-D subspace (K-lines) clustering is very important due to the intriguing relation it exhibits to the problem of dictionary learning. Though, K-lines clustering has been successfully used in sparse component analysis [23], its characteristics have not been studied in literature. For any clustering algorithm, it is essential to analyze its local optimality and convergence for a given set of training data. Furthermore, analyzing the stability characteristics of K-lines clustering will be beneficial in understanding the behavior of dictionary learning. Since the complexity of computing 1-D subspaces in higher dimensions is high, there is a need to build dimensionality reduction schemes that preserve the 1-D subspaces representing the underlying clusters. When the separability between the classes is inadequate, clustering performance can be improved by mapping the input data into a feature space using a non-linear transformation such that the clusters become more separable [33].

1.6.2 Discriminative Clustering for Mixing Matrix Estimation

Clustering algorithms can be combined with dimensionality reduction strategies such as linear discriminant analysis (LDA), in order to improve the separability of the

data belonging to different classes. This motivates the design of discriminative 1-D subspace clustering algorithms in both the input space and the feature space. In addition to providing improved clusterings, these methods will be particularly useful for mixing matrix estimation in blind source separation. Blind source separation (BSS) is the problem of estimating the original source signals from their mixtures, when the mixing process is unknown. All source separation techniques require a reliable estimate of the mixing process, and this can be efficiently performed using discriminative 1-D subspace clustering.

1.6.3 Learning Stable and Generalizable Dictionaries

Inferring data-driven sparse models from a set of training data requires the understanding of stability and generalization characteristics of the learning algorithm. Since learning is performed only with a finite number of samples, the asymptotic convergence of approximation error obtained with these empirical methods to the expected error is crucial for their performance on novel test samples. Furthermore, for the learning to be asymptotically stable, it must depend only on the underlying probability distribution of the sample space, and not on the training samples considered. To design such an a algorithm, the relation between dictionary optimization and 1-D subspace clustering needs to be exploited. In addition, the use of ensemble learning methods, that combine multiple weak hypotheses to sparsely approximate the data samples, can improve the generalization performance.

1.6.4 Multiple Kernel Sparse Representations

The inability of sparse methods to model non-linear relationship between samples, and to combine multiple diverse image descriptors have challenged their use in complex visual recognition. Kernel methods provide a flexible way to learn non-linear models by performing a non-linear transformation (implicit) of the data samples to a feature space, and learning linear models in the resulting space. By building kernel

sparse models, the use of multiple features and incorporation of appropriate distance functions (possibly non-linear) for each feature becomes feasible.

1.6.5 Tumor Segmentation using Kernel Sparse Models

Since kernel sparse models allow the use of diverse features, they are suitable for image segmentation. As an important application, kernel sparse codes can be used to perform brain tumor detection and segmentation. A robust method to automatically segment and identify tumor regions in medical images is extremely valuable for clinical diagnosis and disease modeling. Hence, there is a need to develop an algorithm using kernel sparse models that can work without user intervention, and at moderate computational complexity.

1.6.6 Discriminative Embedding using ℓ_1 Graphs

Several linear and non-linear dimensionality reduction schemes can be unified under the framework of graph embedding. Though classical graph construction approaches have been successful, it has been shown that using sparse codes to establish relation between data samples (ℓ_1 graphs) leads to more robust graphs [34]. This graph construction procedure is unsupervised, and hence cannot be applied to obtain a discriminative embedding. Developing an algorithm that uses ℓ_1 graphs to compute an embedding, which discriminates different classes, will be very useful in medical image segmentation. In particular, this method can be applied for obtaining the glomerular count in kidney MRI images.

1.6.7 Local Sparse Models for Object Recognition and Image Retrieval

An important class of models, referred to as local sparse models, incorporate locality constraints to encourage the code to have non-zero coefficients for dictionary atoms in the neighborhood of the encoded data. Though dictionaries constructed using simple clustering procedures have been used in practice, building data-adapted dictionaries will improve the performance of local sparse coding in object recognition. Further-

more, incorporating supervisory label information into the local sparse model will enable its use in applications such as image retrieval.

1.7 Contributions

In Chapter 3, the theoretical characteristics and implementation aspects of 1-D subspace clustering are analyzed. The convergence of the 1-D subspace clustering algorithm is shown, and it is proved that the algorithm provides a locally optimal solution for a given set of training data, based on Lloyd's optimality conditions. Furthermore, the local optimality is shown by developing an Expectation-Maximization procedure for learning dictionaries to be used in sparse representations, and by deriving the clustering algorithm as its special case. The cluster centroids obtained from the algorithm are proved to tessellate the space into convex Voronoi regions. The stability of clustering is shown by posing the problem as an empirical risk minimization procedure over a function class. It is proved that, under certain conditions, the cluster centroids learned from two sets of i.i.d. training samples drawn from the same probability space become arbitrarily close to each other, as the number of training samples increase asymptotically. A dimensionality reduction procedure, based on random projections and Gaussian Mixture Model (GMM) assumptions, that preserves the 1-D subspaces is developed. Finally, the kernel K-lines clustering algorithm is proposed to perform 1-D subspace clustering in a feature space using kernel methods. Method and results of the analysis described in this chapter have been published in [35], [36].

In Chapter 4, the problem of estimating the mixing matrix in instantaneous blind source separation (BSS) is considered. A novel discriminative 1-D subspace clustering method is proposed for estimating the mixing matrix. The proposed method works by iteratively identifying projections to discriminate the observations from different sources, and computing the mixing parameters using 1-D subspace clustering. In addition to relaxing the key conditions on source sparsity and disjointedness, this

technique can work consistently across different mixing conditions: (a) overdetermined (# sensors > # sources), (b) fully determined (# sensors = # sources), and (c) underdetermined (# sensors < # sources). For cases (a) and (b), the algorithm combines linear discriminant analysis based on similarity measures and 1-D subspace clustering to accurately estimate the mixing matrix. For case (c), estimation is performed in a feature space using kernel methods. Robust estimation of the number of sources is also performed using the proposed technique. Finally, it is demonstrated that the proposed algorithm achieves improved estimation performance when compared to other baseline algorithms. Algorithms and experiments reported in this chapter can be found in [37], [38].

In Chapter 5, the multilevel dictionary (MLD) learning algorithm to design global dictionaries for representing image patches is described. It is shown that, for a sufficient number of levels, the proposed algorithm converges and for a sufficient number of atoms per level the multilevel dictionary captures the energy hierarchy in image patches. Furthermore, a Regularized Multilevel OMP (RM-OMP) procedure is developed for computing the sparse codes of test data using the proposed dictionary. Using the fact that the K-lines clustering algorithm is stable, stability analysis of the MLD algorithm is carried out. It is experimentally demonstrated that, the stability of multilevel learning improves as the difference between their corresponding training sets becomes small and as the number of training samples increases. In addition, it is shown that the error in sparse approximation for the training and test data sets become comparable as the size of the training set increases. In order to improve the generalization of MLD, multiple dictionaries are constructed in each level using random subsets of the training data and the individual approximations are aggregated. Finally, the proposed dictionaries are evaluated in compressed recovery and image compression. Parts of this work have been published in [39], [40].

In Chapter 6, the idea of multiple kernel sparse representations (MKSRs) is introduced and their use in object recognition and unsupervised clustering is described. Two different approaches are developed for obtaining sparse codes and optimizing dictionaries in the unified feature space obtained using multiple kernels. Both the approaches require the extraction of appropriate features, and computation of their corresponding base kernels. The first approach works by building separate dictionaries to sparsely code each descriptor set, and fusing the corresponding kernel matrices to obtain multiple kernel sparse representations. Though the fused kernel matrices are used for evaluating MKSRs, each dictionary is optimized separately using a fixed point algorithm. In the second approach for obtaining MKSRs, kernel dictionary learning is directly performed using the ensemble kernel matrix, constructed by fusing the kernel similarities of all descriptors. In order to obtain kernel dictionaries, we present the kernel multilevel dictionary learning (KMLD) algorithm and design a greedy procedure to obtain sparse codes using the learned dictionary. Finally, the improved recognition and clustering performances of the proposed methods are demonstrated using standard databases. The contents of this chapter have been reported in [41].

In Chapter 7, the use of multiple kernel sparse models in tumor segmentation is addressed. Approaches to automatically segment enhancing/active and necrotic tumor components from T1-weighted contrast-enhanced MR images are developed. By computing kernel sparse codes for the pixels in the image, pixel-based segmentation is performed. The proposed algorithm for localizing the active tumor regions uses an ensemble kernel constructed using pixel intensities and their spatial locations. Each pixel is classified as belonging to a tumor or a non-tumor region using a linear SVM on the kernel sparse codes. In addition, a semi-automated segmentation technique is developed for improved computational efficiency, wherein the user can initialize the tumor region. The proposed algorithms are evaluated on a set of T1-weighted

contrast-enhanced MR images and the results are compared with manual segmentation performed by an expert radiologist. The algorithms and experimental results in this chapter can be found in [42], [43].

In Chapter 8, the problem of computing an discriminative embedding for data samples, using graphs constructed with sparse codes is considered. In particular, this algorithm can be effective for measuring glomerular number in kidney MRI images. Though the general problem of image segmentation can be unsupervised, some prior knowledge about the MRI image intensities at glomeruli locations can be incorporated. This prior knowledge is obtained by manually marking glomeruli regions in a few ground truth images. By learning a suitable discriminative model from labelled training data, the complexity of segmenting a test image is also reduced significantly, in addition to producing accurate results. The proposed approach works by extracting small image patches, and computing a low-dimensional embedding for the patches, such that glomeruli regions are discriminated from the other regions in the MRI. For a test image, the extracted patches are projected onto the discriminant directions and passed to a clustering algorithm to identify the glomeruli. Description of the proposed method and simulation results have been reported in [44].

In Chapter 9, the application of local sparse models in object recognition and image retrieval is explored. An algorithm to design dictionaries for local sparse coding of image descriptors is proposed. In addition, the local sparse coding models are learned in the feature space using the kernel trick. Simulation results for object recognition demonstrate that the two proposed algorithms achieve higher classification accuracies. In order to perform image retrieval using sparse methods, a supervised local sparse coding approach using sub-image heterogeneous features is presented. By performing image retrieval on standard datasets, it is shown that incorporating supervised information into local sparse coding results in improved precision-recall rates. Parts of this work have been reported in [45], [46], and [47].

Chapter 2

SPARSE REPRESENTATIONS AND DICTIONARY LEARNING

Analysis-Synthesis applications in signal and image processing aim to represent data (signals/images) in the most parsimonious terms. In signal analysis, linear models that represent the signals sparsely in a transform domain, such as the Fourier [48] or the wavelet domain [49], have been considered. The general problem of sparse representation is to approximate an input signal using a linear combination of elementary signals. A common metric considered for measuring sparsity of the linear combination is to use the number of elementary signals that participate in the approximation. The nature of the signal determines the suitable transform to be applied, such that a sparse representation could be obtained. For example, the wavelet transform has been successfully used for 1-D signals [49] and curvelets have been found to be optimal for representing the edges in an image [50]. However, analysis of the diverse phenomena that generate signals in practice show that, a single transform cannot be used to describe the signals effectively. As a result, models that use combinations of elementary signals from several different transforms have been considered [51], [52]. In many modern applications, the elementary signals are drawn from a large, linearly dependent collection of signals [53]. These linear models are often referred to as redundant or overcomplete.

The strategy proposed by Olshausen and Field [18] to reduce higher-order redundancy in images is based on using a probabilistic model to capture the image structure. They demonstrated that, learning sparse linear codes for natural images develops a family of localized, oriented and bandpass features, similar to those found in the primary visual cortex. In this model, images are described in terms of a linear superposition of basis functions and these functions are adapted in terms of a collection of statistically independent events. The statistical structure

of naturally occurring signals and images allows for their efficient representation as a sparse linear combination of patterns such as edges, lines and other elementary features [17]. A finite collection of normalized features is referred to as a dictionary. In redundant/overcomplete models, the number of basis functions is greater than the dimensionality of the input signals. It has been shown that, approximation power of the model is significantly improved when an overcomplete set of basis functions are used [54], [55]. The other advantages of using redundant representations are that, they are well behaved in presence of noise and they aid in obtaining shift-invariant representations [56].

2.1 Sparsity Regularization

Considering the generative model for sparse coding (8.9), the codes can be obtained either by minimizing the exact ℓ_0 penalty or its convex surrogate ℓ_1 penalty as,

$$\text{(PL0)}\ \hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmin}} \|\mathbf{a}\|_0 \text{ subj. to } \mathbf{y} = \mathbf{\Phi a}, \tag{2.1}$$

$$\text{(PL1)}\ \hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmin}} \|\mathbf{a}\|_1 \text{ subj. to } \mathbf{y} = \mathbf{\Phi a}, \tag{2.2}$$

where $\|.\|_0$ is the ℓ_0 norm and $\|.\|_1$ is the ℓ_1 norm. Since real-world data cannot be expressed exactly using the generative model in (8.9), usually the equality constraints in (8.10) and (8.11) are replaced using the constraint $\|\mathbf{y} - \mathbf{\Phi a}\|_2^2 \leq \epsilon$, where ϵ is the error goal of the representation. The exact ℓ_0 minimization given in (8.10) is a combinatorial problem and that is the major reason why its convex surrogate is often used. In fact, any penalty function from the set $\{\|\mathbf{a}\|_p \mid 0 < p \leq 1\}$, can be shown to be promote sparsity. The shape of the unit ℓ_p balls which are level sets defined by

$$\left(\sum_{i=1}^{K} |a[i]|^p \right)^{1/p} = 1 \tag{2.3}$$

are shown in Figure 2.1 for various values of p. Note that the $p = 0$ cannot be used in (2.3), since the ℓ_0 norm, that counts the number of non-zero coefficients, is only a pseudonorm. The optimization given in (8.11) can be visualized as the

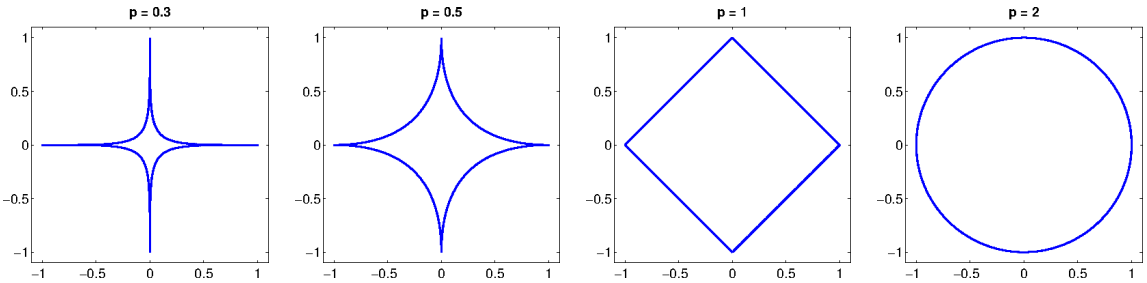


Figure 2.1: Unit ℓ_p balls for $p = 0.3, 0.5, 1, 2$. Note that the only ℓ_p ball that is sparsity promoting and convex is the ℓ_1

expansion the ℓ_1 ball until it touches the affine feasible set $\mathbf{y} = \mathbf{\Phi a}$. Considering the various unit balls in Figure 2.1, it can be shown that all points in the ℓ_2 ball have an equal probability of touching an arbitrary affine feasible set, and hence the solution is almost always dense. However, the balls with $p \leq 1$ have a high probability of touching the feasible set at points where most of the coordinates are zero, leading to sparse solutions with high probability. In the rest of this chapter, we restrict our discussion to ℓ_0 and ℓ_1 norms.

The generative model indicated in (8.9) with sparsity constraints is a non-linear model, because set of all $S-$sparse vectors is not closed under addition. The sum of two $S-$ sparse vectors generally results in a $2S-$sparse vector. Furthermore, sparse models are generalizations of linear subspace models since they each sparse pattern represents a subspace, and the union of all patterns represent a union of subspaces. Considering $S-$sparse coefficient vectors obtained from a dictionary of size $M \times K$, the data samples $\mathbf{y}$ obtained using the model (8.9) lie in a union of $\binom{K}{S}$ $S-$dimensional subspaces.

2.2 Uniqueness of ℓ_0 and its Equivalence to ℓ_1 Solution

So far, we have discussed in length about sparse representations and obtaining representations using the ℓ_0 minimization. However, it is also important to ensure that for a given dictionary $\mathbf{D}$ such a representation obtained is unique, both the ℓ_0 and ℓ_1 solutions obtained using (8.10) and (8.11) are equivalent to each other [51, 57].

To analyze the uniqueness of the solution for an arbitrary (in this case over-complete) dictionary $\mathbf{\Phi}$, assume that there are two suitable representations for the input signal $\mathbf{y}$,

$$\exists \mathbf{a}_1 \neq \mathbf{a}_2 \quad \text{such that} \quad \mathbf{y} = \mathbf{\Phi}\mathbf{a}_1 = \mathbf{\Phi}\mathbf{a}_2. \tag{2.4}$$

Hence the difference of the representations $\mathbf{a}_1 - \mathbf{a}_2$, must be in the null space of the representation, $\mathbf{\Phi}(\mathbf{a}_1 - \mathbf{a}_2) = 0$. This implies that some group of elements in the dictionary should be linearly dependent. To quantify the relation, we define the *spark* of a matrix. Given a matrix, the spark is defined as the smallest number of columns that are linearly dependent. This is quite different from *rank* of a matrix, which is the largest number of columns that are linearly independent. If a signal has two different representations as in (2.4), we must have $\|\mathbf{a}_1\|_0 + \|\mathbf{a}_2\|_0 \geq Spark(\mathbf{\Phi})$. From this argument, if any representation exists satisfying the relation $\|\mathbf{a}_1\|_0 < Spark(\mathbf{\Phi})/2$, then any other representation for the signal $\|\mathbf{a}_2\|_0 > Spark(\mathbf{\Phi})/2$. This indicates that the sparsest representation is $\mathbf{a}_1$. To conclude, a representation is the sparsest possible if $\|\mathbf{a}\|_0 < Spark(\mathbf{\Phi})/2$. Assuming that the dictionary atoms are normalized to unit ℓ_2 norm, let us define the Gram matrix for the dictionary as $\mathbf{G} = \mathbf{D}^T\mathbf{D}$ and denote the coherence as the maximum magnitude of the off-diagonal elements

$$\mu = \max_{i \neq j} |g_{i,j}|. \tag{2.5}$$

Then we have the bound $Spark(\mathbf{\Phi}) > 1/M$ [51, Thm. 5], and hence it can be inferred that the representation obtained from ℓ_0 minimization is unique and equivalent to ℓ_1 minimization if

$$\|\mathbf{a}\|_0 \leq \frac{1}{2}\left(1 + \frac{1}{\mu}\right). \tag{2.6}$$

This is referred to as the *deterministic sparsity threshold*, since it holds true for all sparsity patterns and non-zero values in the coefficient vectors. This threshold is illustrated in Figure 2.2 for various values of μ. Tthe threshold is the same for ℓ_1

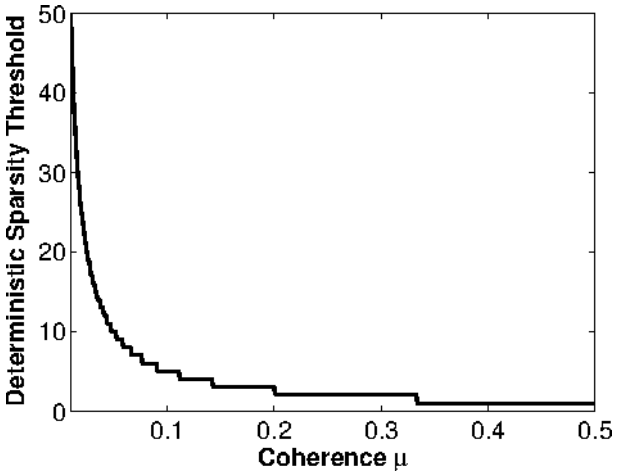


Figure 2.2: Deterministic sparsity threshold with respect to the coherence of the dictionary.

minimization as well as greedy recovery algorithms such as the Orthogonal Matching Pursuit (OMP). The deterministic sparsity threshold scale at best as $sqrtM$ as M increases. Probabilistic or Robust sparsity thresholds, on the other hand scale in the order of $M/\log K$ [58] and break the square-root bottleneck. However, the trade-off is that the unique recovery using ℓ_1 minimization is only assured with high probability, and robust sparsity thresholds for unique recovery using OMP are also unknown.

2.3 Numerical Methods for Sparse Coding

When ℓ_0 norm is used as the cost function, exact determination of the sparsest representation is a NP-hard problem [59], and complexity of the search grows exponentially with the dictionary size. Hence, a number of numerical algorithms that use ℓ_1 approximation and greedy procedures have been developed to solve these problems. Note that most of these algorithms have a non-negative counterpart and it is straightforward to develop them by appropriately placing the non-negativity constraint.

Some of the widely used methods for computing sparse representations include greedy sequential methods such as the Matching Pursuit (MP) [60] and Orthogonal Matching Pursuit (OMP) [61], Basis Pursuit (BP) [62], FOCUSS [63], Feature Sign

Search (FSS) [64], Least Angle Regression (LARS) [65], and iterated shrinkage algorithms [66, 67]. Before describing the sparse coding algorithms, we will present an overview of the optimality conditions used by the procedures.

2.3.1 Optimality conditions

Considering the ℓ_0 optimization problem in (8.10), it can be shown that a unique and hence an optimal solution can be obtained using BP, MP, and OMP algorithms [68] if the condition given in (2.6) is satisfied. For algorithms that use the penalized ℓ_1 formulation given in (8.11), the optimality condition is obtained computing the following sub-gradient set

$$2\mathbf{\Phi}^T(\mathbf{\Phi a} - \mathbf{y}) + \lambda \mathbf{r}, \tag{2.7}$$

and ensuring that it contains the zero vector. Here $\mathbf{r}$ is the sub-gradient of $\|\mathbf{a}\|_1$ and is defined as

$$r[i] = \begin{cases} +1 & a[i] > 0 \\ [-1, +1] & a[i] = 0. \\ -1 & a[i] < 0 \end{cases} \tag{2.8}$$

Hence the optimality conditions can be simplified as [69]

$$2\boldsymbol{\phi}_i^T(\mathbf{y} - \mathbf{\Phi a}) = \lambda \text{sign}(a[i]) \text{ if } a[i] \neq 0, \tag{2.9}$$

$$2|\boldsymbol{\phi}_i^T(\mathbf{y} - \mathbf{\Phi a})| < \lambda \text{ otherwise.} \tag{2.10}$$

These conditions are used as criterion for optimality and coefficient selection by LARS and FSS algorithms.

2.3.2 Basis Pursuit

Basis Pursuit (BP) is a *Linear Programming* (LP) approach [70] that solves (8.11) to find the minimum ℓ_1 norm representation of the signal [62].

A standard linear program is a constrained optimization problem with affine objective and constraints. We can convert the problem in (8.11) to an LP, by adding

an auxiliary variable $\mathbf{u} \in \mathbb{R}^K$ as

$$\min_{\mathbf{u}} \mathbf{1}^T \mathbf{u} \quad \text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{u} \geq 0,\ -\mathbf{u} \leq \mathbf{x} \leq \mathbf{u}. \tag{2.11}$$

Relating the problem in (8.11) to the linear program, the problem is to identify which variables in $\mathbf{x}$ should be zero. To solve this LP, both *simplex* methods and *interior point* methods have been used [62]. Geometrically, the collection of feasible points is a convex polyhedron or a *simplex*. The simplex method works by moving around the boundary of the simplex, jumping from one vertex of the polyhedron to another at which the objective is better. On the contrary, interior point methods start from the interior of the simplex. Since, the solution of LP is at an extreme point of the simplex, as the algorithm converges, the estimate moves towards the boundary. When the representation is not exact and the error goal ϵ is known, the sparse code can be computed by solving the quadratically constrained problem

$$\hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmin}} \|\mathbf{a}\|_1 \text{ subj. to } \|\mathbf{y} - \mathbf{\Phi}\mathbf{a}\|_2 \leq \epsilon. \tag{2.12}$$

This problem is referred to as Basis Pursuit Denoising (BPDN) and several efficient procedures exist for solving this [62]. For a data vector $\mathbf{y}$ of dimensionality M, corrupted with additive white Gaussian noise (AWGN) of variance σ^2, the squared error goal ϵ^2 is fixed at $(1.15\sigma^2)M$. This is because with a probability of 0.75, each component in the AWGN noise vector will lie within the range $[-1.15\sigma, 1.15\sigma]$. By setting this error goal, we will have a very low chance of picking noise as a part of the representation.

2.3.3 Greedy Pursuit Methods

Greedy procedures for computing sparse representations operate by choose the atom that is most strongly correlated to the current target, remove its contribution from it and iterate. Hence, they make a sequence of locally optimal choices in an effort to obtain a global optimal solution. There are several versions of such greedy algorithms, some of which are aggressive and remove all the contribution of the particular

Table 2.1: Matching pursuit algorithm.

Input:
Input signal, $\mathbf{y} \in \mathbb{R}^N$, Dictionary, $\mathbf{\Phi} = \{\phi_k\}_{k \in \mathbf{\Omega}}$.
Output:
Coefficient vector, $\mathbf{a} \in \mathbb{R}^K$.

Initialization:
Initial residual: $\mathbf{r}^{(0)} = \mathbf{y}$,
Initial coefficient vector: $\mathbf{a}^{(0)} = 0$.
Loop index: $J = 1$.

while convergence not reached
- Determine an index $\lambda^{(J)}$: $\lambda^{(J)} \in \underset{k}{\text{argmax}} \, |\langle \mathbf{r}^{(J-1)}, \phi_k \rangle|$.
- Update the coefficient vector: $\mathbf{a}(\lambda^{(J)}) = \mathbf{a}(\lambda^{(J)}) + \langle \mathbf{r}^{(J-1)}, \phi_{\lambda^{(J)}} \rangle$.
- Compute the new residual: $\mathbf{r}^{(J)} = \mathbf{r}^{(J-1)} - \langle \mathbf{r}^{(J-1)}, \phi_{\lambda^{(J)}} \rangle \phi_{\lambda^{(J)}}$.
- Update the loop counter: $J = J + 1$.

end

atom from the target, and some which are less aggressive and remove only a part of the contribution. For orthonormal dictionaries, even the most aggressive greedy algorithms perform well whereas for overcomplete dictionaries, algorithms that are more careful in fixing the coefficient result in a better approximation. The greedy methods for solving sparse approximation problems generalize this simple idea to the case of any arbitrary dictionary. A clear advantage with greedy algorithms is that there is a flexibility to fix the number of non-zero coefficients, and/or the error goal ϵ, which with not the case with BP or BPDN.

Matching Pursuit (MP)

This is the simplest of greedy pursuit algorithms and was proposed by Mallat and Zhang [60]. The steps involved in the MP algorithm are shown in Table 2.1. The algorithm begins by setting the initial residual to the input signal $\mathbf{y}$ and making a trivial initial approximation. It then iteratively chooses the best correlating atom to the residual and updates the residual correspondingly. In every iteration, the algorithm takes into account only the current residual and not the previously selected

Table 2.2: Orthogonal matching pursuit algorithm.

Input:
Input signal, $\mathbf{y} \in \mathbb{R}^N$, Dictionary, $\mathbf{\Phi} = \{\phi_k\}_{k \in \mathbf{\Omega}}$.
Output:
Coefficient vector, $\mathbf{a} \in \mathbb{R}^K$.

Initialization:
Initial residual, $\mathbf{r}^{(0)} = \mathbf{y}$,
Index set, $\mathbf{\Lambda}^{(0)} = \emptyset$.
Loop index, $J = 1$.

while convergence not reached
- Determine an index $\lambda^{(J)}$: $\lambda^{(J)} \in \text{argmax}_k \; |\langle \mathbf{r}^{(J-1)}, \phi_k \rangle|$.
- Update the index set: $\mathbf{\Lambda}^{(J)} = \mathbf{\Lambda}^{(J-1)} \bigcup \lambda^{(J)}$.
- Compute the coefficient $\mathbf{a}$: $\min_{\mathbf{a}} \left\| \mathbf{y} - \sum_{j=1}^{J} \mathbf{a}(\lambda^{(j)}) \phi_{\lambda^{(j)}} \right\|_2$.
- Compute the new residual: $\mathbf{r}^{(J)} = \mathbf{y} - \sum_{j=1}^{J} \mathbf{a}(\lambda^{(j)}) \phi_{\lambda^{(j)}}$.
- Update the loop counter: $J = J + 1$.

end

atoms, thereby making this step greedy. It is important to note that, MP might select the same dictionary atom many times, when the dictionary is not orthogonal. The contribution of an atom to the approximation is the inner product itself as in the case of an orthogonal expansion. The residual is updated by removing the contribution of the component in the direction of the atom $\phi_{\lambda^{(J)}}$. At each step of the algorithm a new approximant of the target signal, $\mathbf{c}^{(J)}$, is calculated based on the relationship

$$\mathbf{c}^{(J)} = \mathbf{y} - \mathbf{r}^{(J)}. \tag{2.13}$$

When the dictionary is orthonormal, the representation $\mathbf{c}^{(S)}$ is always an optimal and unique S-term representation of the signal [61]. It has been shown that, for general dictionaries the norm of the residual converges to zero [71].

Orthogonal Matching Pursuit (OMP)

This algorithm introduces a least squares minimization to each step of the MP algorithm, in order to ensure that the best approximation is obtained over the atoms that have already been chosen [72–74]. This is also referred to as forward selection

algorithm [75] and the steps involved in this algorithm are provided in Table 2.2. The initialization of OMP is similar to that of MP, with a difference that an index set is initialized to hold the indices of atoms chosen at every step. The atom selection procedure of this algorithm is also greedy as in MP. The index set is updated by adding the index of currently chosen atom, to the list of atoms that have already been chosen. Unlike the MP algorithm, where the correlations themselves were the coefficients, OMP computes the coefficients by solving a least squares problem. The most important behavior of OMP is that the greedy selection always picks an atom that is linearly independent from the atoms already chosen. In other words,

$$\langle \mathbf{r}^{(J)}, \phi_{\lambda^{(j)}} \rangle = 0 \quad \text{for } j = 1, ..., J. \tag{2.14}$$

As a result, the residual must be equal to zero in N steps. Further, the index set $\mathbf{\Lambda}$ is of full-rank and hence the least squares solution is unique. Since, the residual is orthogonal to the atoms already chosen, OMP ensures that an atom is not chosen more than once in the approximation. The solution of the least squares problem determines the approximant,

$$\mathbf{c}^{(J)} = \sum_{j=1}^{J} \mathbf{a}(\lambda^{(j)}) \phi_{\lambda^{(j)}}. \tag{2.15}$$

A non-negative version of the OMP algorithm, has been proposed in [76] as a greedy procedure for recovering non-negative sparse representations. Several sophisticated pursuit methods have been proposed recently in order to improve the performance of OMP. One such procedure, the compressive sampling matching pursuit (CoSaMP) [77], follows a procedure similar to the OMP, but selects multiple columns of the dictionary and prunes the set of active columns in each step.

Least Angle Regression (LARS)

The LARS procedure computes the solution to penalized ℓ_1 optimization in (8.11) by selecting the active coefficients one at a time and performing a least squares procedure

at every step to re-estimate them. This method is closely related to the OMP or the forward selection algorithm, but it is less greedy in reducing the residual error. A closely related method to OMP and LARS is the forward stagewise algorithm, which takes thousands of tiny steps as it moves towards the solution. All these algorithms update their current approximant of the signal by taking a step in the direction of the largest correlation of the dictionary atom with the current approximant

$$\mathbf{c}^{(J+1)} = \mathbf{c}^{(J)} + \epsilon sign(\langle \mathbf{r}^{(J)}, \boldsymbol{\psi}_{\hat{j}} \rangle) \text{ where } \hat{j} = \text{argmax} \, |\langle \mathbf{r}^{(J)}, \boldsymbol{\psi}_{j} \rangle|. \tag{2.16}$$

For MP, the step size ϵ is the same as $|\langle \mathbf{r}^{(J)}, \boldsymbol{\psi}_{\hat{j}} \rangle|$, whereas for OMP the current approximant is re-estimated using a least squares method. The Forward Stagewise procedure is overly careful and fixes ϵ as a small fraction of the MP step size. LARS is a method that strikes a balance in step-size selection.

The idea behind the LARS algorithm can be described as follows. The coefficients are initialized as zero and the dictionary atom $\boldsymbol{\psi}_{j_1}$ that is most correlated with the signal $\mathbf{y}$ is chosen. The largest step possible is taken in the direction of $\boldsymbol{\psi}_{j_1}$ until another dictionary atom $\boldsymbol{\psi}_{j_2}$ has as much correlation with the current residual. Then the algorithm proceeds in the equiangular direction between $\boldsymbol{\psi}_{j_1}$ and $\boldsymbol{\psi}_{j_2}$ until a new dictionary atom $\boldsymbol{\psi}_{j_3}$ has as much correlation with the residual as the other two. Therefore, at any step in the process LARS proceeds equiangularly between the currently chosen atoms. The difference between MP and LARS is that in the former case, a step is taken in the direction of the atom with maximum correlation with the residual, whereas in the latter case the step taken is in the direction that is equiangular to all the atoms that are most correlated with the residual. Note that the LARS algorithm always adds coefficients to the active set and never removes any. This could lead us to think that it may not result in an optimal solution for (8.11) and that is true. Considering the optimality constraints in (2.9) and (2.10), the LARS algorithm implictly checks for the all the conditions except the sign condition in (2.9).

Hence, in order to make the LARS solution the same as optimal solution for (8.11), a small modification that provides a way for deletion of coefficients from the current non-zero coefficient set has to be included.

The LARS algorithm can produce a whole set of solutions for (8.11) for λ varying between 0, when M dictionary atoms are chosen and a least squares solution is obtained, to its original value, where a sparse solution is obtained. If the computations are arranged properly, LARS is a computationally cheap procedure with complexity of the same order as that of least squares. The dictionary is of size 10×100 whose entries are obtained from $\mathcal{N}(0, 1)$, and the non-zero entries in the coefficient vector are obtained from a uniform random distribution.

2.3.4 Feature-Sign Search

Feature sign search is an efficient sparse coding method that solves the penalized ℓ_1 optimization in (8.11) by maintaining an *active set* of non-zero coefficients and corresponding signs, and searches for the optimal active set and coefficient signs. In each *feature-sign step*, given an active set, an analytical solution is computed for the resulting quadratic program. Then, the active set and the signs are updated using a discrete line search and by selecting coefficients that promote optimality of the objective. The sign vector is denoted by $\boldsymbol{\theta}$ and its entries are defined as $\theta[i] = sign(x[i])$, which is one of the elements from $\{-1, 0, +1\}$. The algorithm consists of the following steps.

Step 1: An element of the non-active set that results in the maximum change in the error term is chosen as

$$i = \underset{i}{\operatorname{argmax}} \left| \frac{\partial \|\mathbf{x} - \mathbf{\Psi a}\|}{\partial a[i]} \right|, \tag{2.17}$$

and add i to the active set if it improves the objective locally. This is obtained when the sub-gradient given by (2.7) is less than 0 for a given index i. Hence we append i to $\mathcal{A}$, if $\frac{\partial \|\mathbf{x}-\mathbf{\Psi a}\|}{\partial a[i]} > \lambda/2$ setting $\theta[i] = -1$, or if $\frac{\partial \|\mathbf{x}-\mathbf{\Psi a}\|}{\partial a[i]} < -\lambda/2$ setting $\theta[i] = 1$.

Step 2: Using coefficients only from the active set, let $\hat{\mathbf{\Phi}}$ be the sub-dictionary corresponding to the active set, and let $\hat{\mathbf{a}}$, $\hat{\boldsymbol{\theta}}$ be the active coefficient and sign vectors. The strategy used in this Feature-Sign step is to find the new set of coefficients that have the sign pattern consistent with the sign vector. In order to do this, we first solve the unconstrained optimization,

$$\min_{\hat{\mathbf{a}}} \|\mathbf{x} - \hat{\mathbf{\Psi}}\hat{\mathbf{a}}\|_2^2 + \frac{\lambda}{2}\hat{\boldsymbol{\theta}}^T\hat{\mathbf{a}} \tag{2.18}$$

and obtain $\hat{\mathbf{a}}_{new}$. Denoting the objective in (2.18) as $f(\hat{\mathbf{a}}, \hat{\boldsymbol{\theta}})$, it is easy to see that $f(\hat{\mathbf{a}}_{new}, \hat{\boldsymbol{\theta}}) < f(\hat{\mathbf{a}}, \hat{\boldsymbol{\theta}})$ since $\hat{\mathbf{a}}_{new}$ is the optimal solution of (2.18). If $sign(\hat{\mathbf{a}}_{new}) = \hat{\boldsymbol{\theta}}$, $\hat{\mathbf{a}}_{new}$ is the updated solution. Else, a line search will be performed between $\hat{\mathbf{a}}$ and $\hat{\mathbf{a}}_{new}$, and the points where at least one coefficient in the vector becomes zero are noted. Among these, the one with the lowest objective value, $\hat{\mathbf{a}}_l$ is chosen as the new coefficient vector. Because of the convexity of f, we have $f(\hat{\mathbf{a}}_l, \hat{\boldsymbol{\theta}}) < f(\hat{\mathbf{a}}, \hat{\boldsymbol{\theta}})$, and hence the objective value decreases. In this case, elements are removed from the active set since at least one of the coefficients in the active set has become zero, and $\boldsymbol{\theta} = sign(\mathbf{a})$.

Step 3: Optimality conditions need to be checked for both non-zero and zero coefficients. For non-zero coefficients belonging to the set $\mathcal{A}$, the condition is given by (2.9). If this does not hold true, *Step 2* is repeated without any new activation. For zero coefficients, the condition is given by (2.10) is checked and if this is not true, the algorithm proceeds to *Step 1*. However, if both optimality conditions are satisfied the algorithm exits with the current solution as optimal.

It can be seen that the feature sign step (Step 2) of the algorithm strictly reduces the objective, and Step 3 of the algorithm mandates that the iterative procedure stops only when optimal solution is attained. It can be shown using a simple proof that the FSS algorithm produces a globally optimal solution and more details on this can be found in [64].

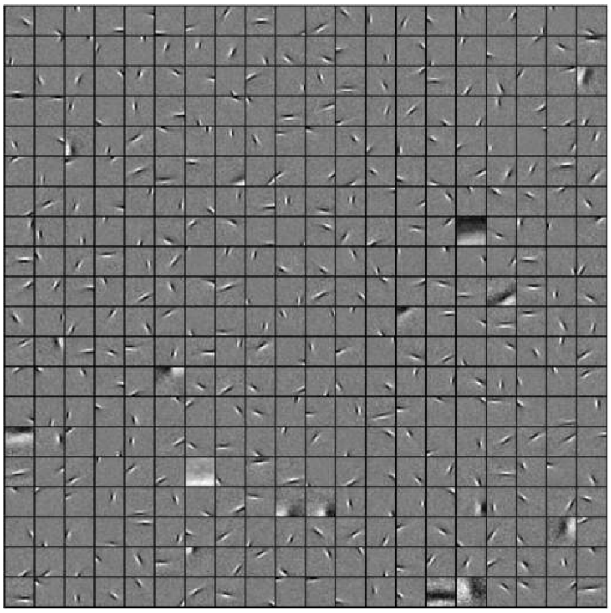


Figure 2.3: An example dictionary learned using the algorithm proposed by Olshausen and Field [2].

2.4 The Art of Dictionary Design

Dictionaries used for sparse representations can be constructed based on the mathematical model that generates the data. There are also methods available to tune the parameters of a pre-defined dictionary, such that the performance over the given set of data is optimized. However, dictionaries that are learned directly from the data result in an improved performance compared to both pre-defined as well as tuned dictionaries. This chapter will focus exclusively on learned dictionaries and their applications in various image processing tasks.

Pre-defined dictionaries can be constructed using bases from transforms such as the discrete cosine transform (DCT), wavelet, curvelet [78] and they have been successfully in various image reconstruction applications. In addition, pre-defined dictionaries can also be constituted as a union of multiple basis, where each basis represents certain features in the image well [79, 80], and this generally results in improved performances compared to using a single basis. A well-known example

for tunable dictionaries are wavelet packets, which generates a library of bases for a given wavelet function, and the basis that results in the optimal performance for the data can be chosen [81]. In their celebrated work, Olshausen and Field proposed a framework to learn an overcomplete set of individual vectors optimized to the training data for sparse coding [18]. Since then, a wide range of dictionary learning algorithms for sparse coding have been proposed in the literature, some of which are tailored for specific applications. In addition to optimizing dictionaries for the data, some applications have directly used training examples as the dictionary. Such models are referred to as *example-based sparse models* [82]. A few important algorithms for learned dictionaries that are useful in general signal and image processing applications are discussed in this chapter.

Assume that the training data $\mathbf{x}$ is obtained from a probability space obeying the generative model posed in (8.9), the dictionary learning problem can be expressed as minimizing the objective

$$g(\mathbf{D}) = \mathbf{E}_{\mathbf{x}}[h(\mathbf{x}, \mathbf{D})] \tag{2.19}$$

where the columns of $\mathbf{D}$, referred to as dictionary atoms, are constrained as $\|\mathbf{d}_j\|_2 \leq 1, \forall j$. The cost of the penalized ℓ_1 minimization given in (8.11) is denoted as $h(\mathbf{x}, \mathbf{D})$. If the continuous probability distribution is unknown and we only have T training samples $\{\mathbf{x}_i\}_{i=1}^{L}$, equiprobable with a mass of $\frac{1}{T}$, (2.19) can be modified as the empirical cost function,

$$\hat{g}(\mathbf{D}) = \frac{1}{T} \sum_{i=1}^{T} h(\mathbf{x}_i, \mathbf{D}). \tag{2.20}$$

Typically dictionary learning algorithms solve for the sparse codes and obtain the dictionary by minimizing $\hat{g}(\mathbf{D})$, repeating the steps until convergence [83–85].

Given the sparse codes dictionary learning can be posed as the convex problem

$$\min_{\mathbf{D}} \sum_{i=1}^{T} \|\mathbf{x}_i - \mathbf{D}\mathbf{a}_i\|_2^2 \text{ subj. to } \|\mathbf{d}_j\|_2 \leq 1, \forall j, \tag{2.21}$$

where $\mathbf{D} = [\mathbf{d}_1\ \mathbf{d}_2\ \ldots\ \mathbf{d}_K]$ is the dictionary matrix and S is the sparsity of the coefficient vector. Denoting $\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2\ \ldots\ \mathbf{x}_T]$ as the collection of T training vectors and $\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2\ \ldots\ \mathbf{a}_T]$ as the coefficient matrix, the objective in (2.21) can be re-written as $\|\mathbf{X} - \mathbf{DA}\|_F^2$ where $\|.\|_F$ denotes the Frobenius norm. Learned dictionaries have been successfully applied in image deblurring, compression, denoising, inpainting and super-resolution [20].

2.5 Dictionary Learning Algorithms

2.5.1 Method of Optimal Directions

This algorithm was proposed by Engan *et.al.* [84], [85], and is one of the early efforts for building learned dictionaries. The procedure optimizes (2.20) using alternating minimization, and in the first step, the current estimate of the dictionary is used to solve for the sparse codes using the ℓ_0 optimization

$$\min_{\mathbf{a}_i} \|\mathbf{x}_i - \mathbf{Da}_i\|_2^2 \quad \text{subject to} \quad \forall i, \quad \|\mathbf{a}_i\|_0 \leq S, i = \{1, 2, \ldots, T\}, \tag{2.22}$$

which can be solved using algorithms such as OMP or FOCUSS [86], [87].

In the second step, the dictionary in the optimization is updated assuming that the sparse codes are known. For the dictionary update step, we define the errors $\mathbf{e}_i = \mathbf{x}_i - \mathbf{Da}_i$. Hence the MSE of the overall representation is given by

$$\text{MSE} = \frac{1}{T}\|\mathbf{X} - \mathbf{DA}\|_F^2. \tag{2.23}$$

The algorithm attempts to update $\mathbf{D}$ such that the above error is minimized. Differentiating (2.23) with respect to $\mathbf{D}$, we obtain $(\mathbf{X} - \mathbf{DA})\mathbf{A}^T = 0$. The dictionary update step can be expressed as

$$\mathbf{D}^{(l+1)} = \mathbf{X}\mathbf{A}^{(l)^T}(\mathbf{A}^{(l)}\mathbf{A}^{(l)^T})^{-1}, \tag{2.24}$$

where the superscript indicates the iteration number. The columns of the updated dictionary $\mathbf{D}^{(l+1)}$ are then normalized. Note that this is the best possible dictionary

that can be obtained for the known coefficient matrix. Using other update schemes such as iterative steepest descent will result in a much slower learning procedure.

2.5.2 K-SVD

The next algorithm that we will consider is the K-SVD, proposed by Aharon et. al. [83], which is also an alternating optimization procedure that attempts to minimize the objective in (2.20). Similar to the MOD, the dictionary $\mathbf{D}$ is fixed and the best coefficient matrix $\mathbf{A}$ is computed using a pursuit method, during the sparse coding step. The K-SVD algorithm has a markedly different dictionary update step, where it updates one column at a time fixing all columns in the dictionary except one, $\mathbf{d}_k$, and updating it along with its corresponding coefficients such that the MSE is reduced. Modifying the coefficients during the dictionary update step imparts a significant acceleration in learning.

During the dictionary update, we assume that both $\mathbf{D}$ and $\mathbf{A}$ are fixed, except for one column in the dictionary, $\mathbf{d}_k$, and the coefficients corresponding to it, i.e., the i^{th} row in $\mathbf{A}$ denoted by $\mathbf{a}_T^i$. Now, the primary objective function can be simplified as,

$$\begin{aligned} \|\mathbf{X} - \mathbf{DA}\|_F^2 &= \left\| \mathbf{X} - \sum_{j=1}^{K} \mathbf{d}_j \mathbf{a}_T^j \right\|_F^2 \\ &= \left\| \mathbf{X} - \left(\sum_{j \neq k} \mathbf{d}_j \mathbf{a}_T^j \right) - \mathbf{d}_k \mathbf{a}_T^k \right\|_F^2 \\ &= \left\| \mathbf{E}_k - \mathbf{d}_k \mathbf{a}_T^k \right\|_F^2 . \end{aligned} \tag{2.25}$$

It can be seen from (2.25) that the product term $\mathbf{DA}$ has been decomposed into K rank-1 matrices among which $K-1$ terms are fixed. The matrix $\mathbf{E}_k$ indicates the errors for all the T examples when the contribution of the k^{th} atom is removed. The natural idea will be to use the SVD to find the new $\mathbf{d}_k$ and the corresponding $\mathbf{a}_T^k$. The SVD finds the closest rank-1 matrix, in terms of Frobenius norm, that

Table 2.3: K-SVD Algorithm.

Initialization:
Initial dictionary, $\mathbf{D}^{(0)}$, with normalized columns, $l = 1$.
while convergence not reached
Sparse Coding Step:
$\min_{\mathbf{a}_i}\{\|\mathbf{x}_i - \mathbf{D}\mathbf{a}_i\|_2^2\}$ subject to $\|\mathbf{a}_i\|_0 \leq S$, for $i = \{1, \ldots, N\}$
Dictionary Update Step: For each column k in $\mathbf{D}^{(l-1)}$, update it by
- Define $\omega_k = \{i | 1 \leq i \leq K, \mathbf{a}_T^k \neq 0\}$.
- Compute $\mathbf{E}_k = \mathbf{X} - \left(\sum_{j \neq k} \mathbf{d}_j \mathbf{a}_T^j\right)$.
- Compute $\mathbf{E}_R^k$, by selecting only the columns corresponding to ω_k from $\mathbf{E}_k$.
- Apply SVD decomposition $\mathbf{E}_R^k = \mathbf{U}\boldsymbol{\Delta}\mathbf{V}^T$.
- Choose the updated dictionary atom, $\mathbf{d}_k$, as the first column of $\mathbf{U}$.
- Updated coefficient vector as the product of first column of $\mathbf{V}$ and $\boldsymbol{\Delta}(1,1)$.
- Update the loop counter, $l = l + 1$.
end

will approximate $\mathbf{E}_k$ and thereby minimizes the expression in (2.25). However, this ignores the sparsity in $\mathbf{a}_T^k$ and will result in a dense representation.

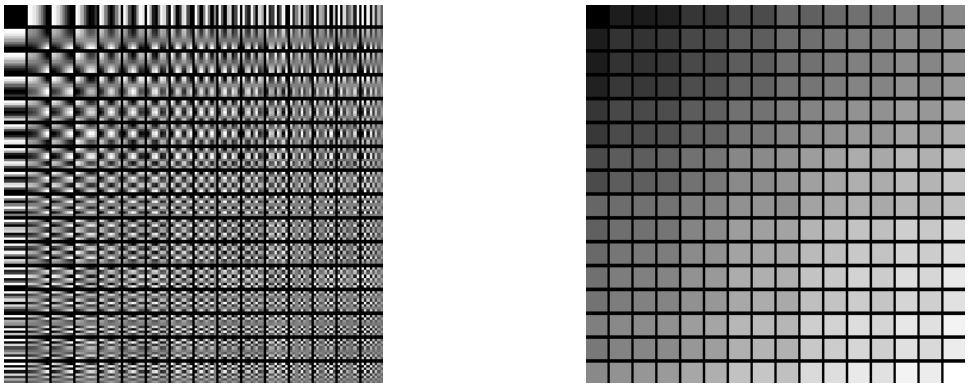


Figure 2.4: An overcomplete DCT dictionary (left) is shown along with the activity measures of its atoms (right), which is low for geometric atoms and high for texture-like atoms.

The K-SVD algorithm handles this issue by defining a set ω_i containing the group of indices, pointing to the columns $\mathbf{x}_i$ of the input signal matrix, that use the atom $\mathbf{d}_k$. In effect,

$$\omega_k = \{i | 1 \leq i \leq K, \mathbf{a}_T^k[i] \neq 0\}. \tag{2.26}$$

Now, define a matrix $\boldsymbol{\Omega}_k$ with ones on the $(\omega_k[i], i)^{\text{th}}$ entries and zeros elsewhere. The

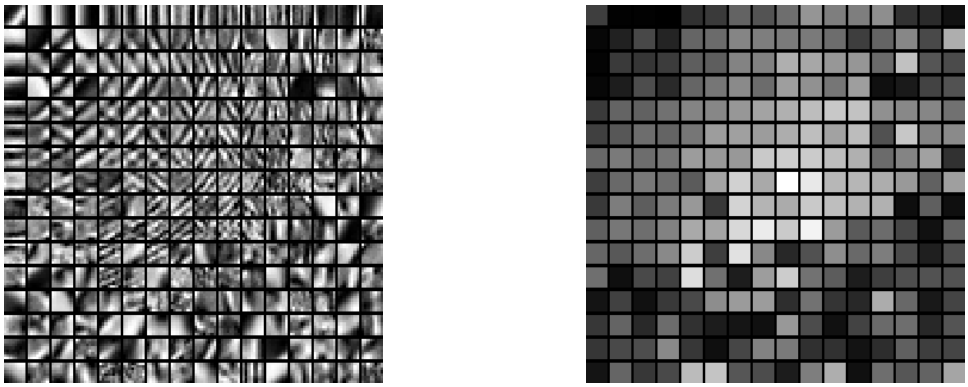


Figure 2.5: An example K-SVD dictionary (left) is shown along with the activity measures of its atoms (right), which is low for geometric atoms and high for texture-like atoms.

vector $\mathbf{a}_R^k = \mathbf{a}_T^k \mathbf{\Omega}_k$ contains only the non-zero entries in $\mathbf{a}_T^k$. Similarly, multiplying $\mathbf{X}_R^k = \mathbf{X}\mathbf{\Omega}_k$ creates a matrix with the subset of the training examples that use the atom $\mathbf{d}_k$. The same happens in $\mathbf{E}_R^k = \mathbf{E}_k\mathbf{\Omega}_k$. This penalty in (2.25) is simplified as,

$$\left\|\mathbf{E}_k\mathbf{\Omega}_k - \mathbf{d}_k\mathbf{a}_T^k\mathbf{\Omega}_k\right\|_F^2 = \left\|\mathbf{E}_R^k - \mathbf{d}_k\mathbf{a}_R^k\right\|_F^2 \tag{2.27}$$

and the minimum of this expression is obtained by updating the dictionary atom $\mathbf{d}_k$ and the coefficient vector $\mathbf{a}_R^k$ using the SVD of $\mathbf{E}_R^k$. The steps involved in this algorithm are formally summarized in Table 2.3. It is shown in [83] that convergence of this algorithm to a local minimum is always guaranteed with a behavior similar to Gauss-Seidel methods in optimization. The predefined DCT and the learned K-SVD dictionaries are compared in Figure 2.4 and Figure 2.5 respectively. The K-SVD dictionary is learned from $50,000$ grayscale patches of size 8×8 extracted from 250 training images of the Berkeley segmentation dataset (BSDS) [1]. Besides the dictionary atoms, a total-variation-like activity measure [20] is also shown for both the dictionaries. The measure for the atom $\mathbf{d}$ organized as a patch of size $\sqrt{M} \times \sqrt{M}$ is defined as

$$\sum_{m=2}^{M}\sum_{n=1}^{M}|d[m,n] - d[m-1,n]| + \sum_{m=1}^{M}\sum_{n=2}^{M}|d[m,n] - d[m,n-1]| \tag{2.28}$$

The measure is high for texture-like patches and low if the patches have a regular geometric structure. The measure is normalized such that its maximum value is 1 and shown for both DCT and K-SVD atoms in Figures 2.4 and 2.5. It can be seen that the DCT dictionary is more regular with respect to this measure compared to the K-SVD dictionary.

2.5.3 Online Dictionary Learning

The dictionary learning algorithms described so far are batch procedures, since they require access to the whole training set in order to minimize the cost function. Hence, the size of the data set that can be used for training is limited. In order for the learning procedure to scale up to millions of training samples, we need an online algorithm that is both efficient both in terms of memory and computations. We will discuss one such procedure proposed by Mairal *et. al.* [88], based on stochastic approximations.

Although dictionary learning results in a solution that minimizes the *empirical cost* g given in (2.20), it will approach the solution that minimizes the *expected cost* $\hat{g}$ given in (2.19), as the number of samples $T \to \infty$. The idea of online learning is to use a well-designed stochastic gradient algorithm that can result in a lower expected cost, when compared to an accurate empirical minimization procedure [89]. Furthermore, for large values of T, empirical minimization of (2.20) using a batch algorithm becomes computationally infeasible and it is necessary to use an online algorithm.

The online algorithm alternates between computing sparse code for the t^{th} training sample, $\mathbf{x}_t$, with the current estimate of the dictionary $\mathbf{D}_{t-1}$, and updating the new dictionary $\mathbf{D}_t$ by minimizing the objective

$$\hat{g}_t(\mathbf{D}) \equiv \frac{1}{t}\sum_{i=1}^{t}\left(\frac{1}{2}\|\mathbf{x}_i - \mathbf{D}\mathbf{a}_i\|_2^2 + \lambda\|\mathbf{a}_i\|_1\right), \tag{2.29}$$

along with the constraint that the columns of $\mathbf{D}$ are of unit ℓ_2 norm. The sparse codes for the samples $i < t$ are used from the previous steps of the algorithm. The online

algorithm for dictionary learning is given in Table 2.4. The dictionary update using warm restart is performed by updating each column of it separately. The detailed algorithm for this update procedure is available in [88]. The algorithm is quite general and can be tuned to certain special cases. For example, when a fixed-size dataset is used, we can randomly cycle through the data multiple times and also remove the contribution of the previous cycles. When the dataset is huge, the computations can be speeded up by processing in mini-batches instead of one sample at a time. In addition, to improve the robustness of the algorithm, unused dictionary atoms must be removed and replaced with a randomly chosen sample from the training set, in a manner similar to clustering procedures. Since the reliability of the dictionary improves over iterations, the initial iterations may be slowed by adjusting the step size, and down-weighting the contributions of the previous data.

In [88], it is proved that this algorithm converges to a stationary point of the objective function. This is shown by proving that under the assumptions of compact support for the data, convexity of the functions $\hat{g}_t$ and uniqueness of the sparse coding solution, $\hat{g}_t$ acts as a converging surrogate of g, as the total number of training samples $T \to \infty$.

2.5.4 Learning Structured Sparse Models

The generative model in (8.9) assumes that the data is generated as a sparse linear combination of dictionary atoms. When solving an inverse problem, the observations are usually a degraded version of the data denoted as

$$\mathbf{z} = \mathbf{U}\mathbf{x} + \mathbf{n}, \tag{2.30}$$

where $\mathbf{U} \in \mathbb{R}^{N \times M}$ is the degradation operator with $N \leq M$, and $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_N)$. Considering the case of images, $\mathbf{x}$ usually denotes an image patch having the sparse representation $\mathbf{D}\boldsymbol{\alpha}$, the matrix $\mathbf{U}$ performs operations such as downsampling, masking, or convolution. Restoration of original images corrupted by these degradations

Table 2.4: Online Dictionary Learning.

Initialization:
- T training samples drawn from the random distribution $p(\mathbf{x})$.
- Initial dictionary, $\mathbf{D}^{(0)} \in \mathbb{R}^{M\times K}$, with normalized columns.
- λ, regularization parameter.
- Set $\mathbf{B} \in \mathbb{R}^{K\times K}$ and $\mathbf{C} \in \mathbb{R}^{M\times K}$ to zero matrices.

for $t = 1$ **to** T
- Draw the training sample $\mathbf{x}_t$ from $p(\mathbf{x})$.
- Sparse Coding:
  $\mathbf{a}_t = \text{argmin}_{\mathbf{a}} \frac{1}{2}\|\mathbf{x}_i - \mathbf{D}_{t-1}\mathbf{a}\|_2^2 + \lambda\|\mathbf{a}\|_1$
- $\mathbf{B}_t = \mathbf{B}_{t-1} + \mathbf{a}_t\mathbf{a}_t^T$
- $\mathbf{C}_t = \mathbf{C}_{t-1} + \mathbf{x}_t\mathbf{a}_t^T$
- Compute $\mathbf{D}_t$ using $\mathbf{D}_{t-1}$ as warm restart,
  $\mathbf{D}_t = \frac{1}{t}\sum_{i=1}^{t}\left(\frac{1}{2}\|\mathbf{x}_i - \mathbf{D}\mathbf{a}_i\|_2^2 + \lambda\|\mathbf{a}_i\|_1\right)$
  $= \frac{1}{t}\sum_{i=1}^{t}\left(\frac{1}{2}\text{Tr}(\mathbf{D}^T\mathbf{D}\mathbf{A}_t) - \text{Tr}(\mathbf{D}^T\mathbf{D}\mathbf{A}_t)\right)$

end for
- Return learned dictionary $\mathbf{D}_T$.

correspond to the inverse problems of single-image superresolution, inpainting and deblurring respectively. The straighforward method of restoring these images, using sparse representations, is to consider **UD** as the dictionary and compute the sparse coefficient vector **a** using the observation **z**. The restored data is then given as **Da**. Apart from the obvious necessity that **x** should be sparsely decomposable in **D**, the conditions to be satisfied in order to get a good reconstruction are [90]: (a) the norms of any column in the dictionary **UD** should not become close to zero, as in this case it is not possible to recover the corresponding coefficient with any confidence, and (b) the columns of **UD** should be incoherent, since coherent atoms lead to ambiguity in coefficient support. For uniform degradations such as downsampling and convolution, even an orthonormal **D** could result in **UD** that violate these two conditions. For example, consider that **U** is an operator that downsamples by a factor of two, in which case the DC atom $\{1, 1, 1, 1, \ldots\}$ and the highest frequency atom $\{1, -1, 1, -1, \ldots\}$ will become identical after downsampling. If the dictionary **D** contains directional filters, an isotropic degradation operator **U** will not lead to a complete loss in the

incoherence property for **UD**. In order to overcome these issues, and obtain a stable representation for inverse problems, it is necessary to consider the fact that similar data admit similar representations and design dictionaries accordingly. This corresponds to the simultaneous sparse approximation (SSA) problem where a set of data will be represented by a set of dictionary atoms. In the section, we will discuss briefly the formulations by Yu *et. al.* [90] and Mairal *et. al.* [91] that are specifically designed for inverse problems in imaging.

Dictionaries for Piecewise Linear Approximation

The main idea in building these dictionaries is to design a set of directional bases, and represent each cluster of image patches linearly using a basis set. Linear modeling using directional basis and simultaneous approximation impart stability to the representation and hence ensures that the restoration performance is high for the degraded data given by (2.30). The patches themselves are modeled as a mixture of C Gaussians, each representing a linear model, i.e., $p(\mathbf{x}_i) = \mathcal{N}(\boldsymbol{\mu}_{c_i}, \boldsymbol{\Sigma}_{c_i})$. An EM-MAP procedure computes the *Maximum-a-Posteriori* (MAP) estimates for the restored patches using an *Expectation-Maximization* (EM) algorithm, by performing the following steps iteratively: (a) given a set of degraded patches $\{\mathbf{z}_i\}_{i=1}^{T}$, the parameters of the C Gaussians are obtained using maximum likelihood estimation, (b) for each patch i, identify the Gaussian that generates it, and (c) estimate each non-degraded patch $\mathbf{x}_i$ from its corresponding Gaussian. Without loss of generality, we can assume that the means $\{\boldsymbol{\mu}_c\}_{c=1}^{C}$ are zero, and hence the only parameter to be estimated for each Gaussian is its covariance $\boldsymbol{\Sigma}_j$. In order to compute the non-degraded patch and its membership with respect to a cluster, we perform the following MAP estimation

$$(\mathbf{x}_i, c_i) = \underset{\mathbf{x},c}{\operatorname{argmax}} \log p(\mathbf{x}|\mathbf{z}_i, \boldsymbol{\Sigma}_c).$$

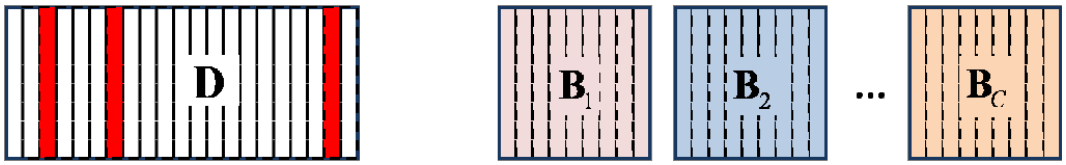


Figure 2.6: Sparse coding with dictionaries (left) and piecewise linear approximation using multiple PCA bases (right).

Using Bayes rule and realizing that $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_M)$, we have

$$(\mathbf{x}_i, c_i) = \underset{\mathbf{x},c}{\operatorname{argmax}} \left(\log p(\mathbf{z}_i|\mathbf{x}, \mathbf{\Sigma}_c) + \log p(\mathbf{x}|\mathbf{\Sigma}_c) \right), \tag{2.31}$$

$$= \underset{\mathbf{x},c}{\operatorname{argmin}} \left(\|\mathbf{U}_i\mathbf{x} - \mathbf{z}_i\|_2^2 + \sigma^2 \mathbf{x}^T \mathbf{\Sigma}_c^{-1} \mathbf{x} + \sigma^2 \log |\mathbf{\Sigma}_c| \right). \tag{2.32}$$

This joint estimation can be divided into computation of per-cluster data $\mathbf{x}_i^c$ and the cluster membership c_i as follows,

$$\mathbf{x}_i^c = \underset{\mathbf{x}}{\operatorname{argmin}} \left(\|\mathbf{U}_i\mathbf{x} - \mathbf{z}_i\|_2^2 + \sigma^2 \mathbf{x}^T \mathbf{\Sigma}_c^{-1} \mathbf{x} \right), \tag{2.33}$$

$$c_i = \underset{c}{\operatorname{argmin}} \left(\|\mathbf{U}_i\mathbf{x}_i^c - \mathbf{z}_i\|_2^2 + \sigma^2 (\mathbf{x}_i^c)^T \mathbf{\Sigma}_c^{-1} \mathbf{x}_i^c + \sigma^2 \log |\mathbf{\Sigma}_c| \right), \tag{2.34}$$

and finally assigning $\mathbf{x}_i = \mathbf{x}_i^c$. Since $\mathbf{\Sigma}_c$ could become rank-deficient, a small constant is added to it in order to regularize it. This procedure is also referred to as Piecewise Linear Estimation (PLE) since each Gaussian in the mixture corresponds to a linear model.

In order to provided a structured sparse model interpretation and to overcome the issue of rank deficiency in $\mathbf{\Sigma}_c$, we perform the eigen decomposition, $\mathbf{\Sigma}_c = \mathbf{B}_c \mathbf{\Lambda}_c \mathbf{B}_c^T$, where $\mathbf{\Lambda}_c$ is a diagonal matrix with the decreasing eigenvalues $\{\lambda_1^c, \ldots \lambda_M^c\}$. The patch belonging to that Gaussian can be conveniently represented using the obtained PCA basis as

$$\mathbf{x}_i^c = \mathbf{B}_c \mathbf{a}_i^c. \tag{2.35}$$

Hence (2.33) transforms to

$$\mathbf{a}_i^c = \underset{\mathbf{a}}{\operatorname{argmin}} \left(\|\mathbf{U}_i\mathbf{B}_c\mathbf{a} - \mathbf{z}_i\|_2^2 + \sigma^2 \sum_{m=1}^{M} \frac{|\mathbf{a}(m)|^2}{\lambda_m^c} \right) \tag{2.36}$$

which is a linear estimation problem on the basis $\mathbf{B}_c$. Note that this is similar to a SSA problem, assuming that the group of dictionary atoms that represent the data are known. The selection of optimal basis for the data $\mathbf{x}_i$ is equivalent to the selection of the best cluster in (2.34). Therefore, the EM-MAP procedure can be implemented by iteratively performing: (a) linear estimation in the C basis sets, each of which corresponds to a cluster, (b) selection of the best basis set for each patch, and (c) estimating the basis sets using the associated patches. The eigenvalues for any cluster fall rapidly, and hence (2.36) leads to a very sparse representation. In actual implementation, (2.36) is never used to estimate the non-degraded patch. Instead, (2.33) and (2.34) are used to perform recovery of the degraded patches.

Since the PLE procedure is non-convex, the algorithm can only converge to a local minimum. Therefore, initialization of the model plays a key role in the ensuring a good estimate of $\mathbf{x}$. The initialization also depends on the degradation operator $\mathbf{U}$, since the recoverability condition that the norms of columns of $\mathbf{UD}$ be greater than zero has to be satisfied. For operators such as downsampling and random masking, the initialization is performed with *directional PCA* basis as follows. Images with oriented edges at different angles are chosen and patches are extracted from different positions in the edge regions. PCA is performed on the extracted patches to compute the basis and the eigen values. They are shown in Figure 2.7, and correspond to Diracs in Fourier space. In practice, the eigen values are computed only once and the eigen vectors are computed using a Gram Schmidt procedure to initialize the covariance matrices. When $\mathbf{U}$ is a blur operator, directional PCA basis cannot be used for initialization since they will not satisfy the recoverability condition. Therefore, we compute the position *PCA basis*, that are spread in Fourier space, by blurring the edge image of a specific orientation with various amounts of blur and extracting the patches from the same positions. After initialization, the multiple iterations of the MAP-EM procedure is used to compute the restored image.

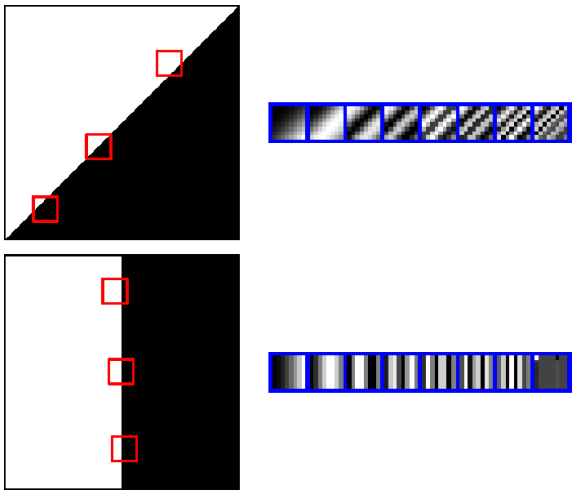


Figure 2.7: Examples for directional PCA basis.

2.6 Use of Sparse Models in Discrimination Tasks

Sparse learning, in addition to providing interpretable and meaningful models, is efficient for large scale learning. Though the paradigm of sparse coding has been very effective in modeling natural signals, its ability to discriminate different classes of data is not inherent. Since sparse coding algorithms aim to reduce only the reconstruction error, they do not explicitly consider the correlation between the codes, which is crucial for classification tasks. As a result, the codes obtained for similar signals or features (extracted from the signals) may be quite different in traditional sparse coding. In addition, the overcomplete nature of the dictionary makes the sparse coding process highly sensitive to the variance in the features. Hence, adapting this representative model to perform discriminative tasks requires the incorporation of supervisory information into the sparse coding and dictionary learning problems. By introducing the prior knowledge on the sparsity of signals into the traditional machine learning algorithms, novel discriminative frameworks can be developed.

2.6.1 Discriminative Dictionary Learning

Sparse representations using predefined and learned dictionaries have been extremely successful in representation of images. However, in order for them to be successful in classification, explicit constraints for discrimination need to be incorporated while adapting the dictionary. It is well known that the traditional linear discriminant analysis (LDA) approach can aid in supervised classification by projecting the data on to a lower dimensional space where the inter-class separation is maximized while the intra-class separation is minimized [92]. If the sparse representation is constrained such that a discriminatory effect can be introduced in the coefficient vectors of different classes, the computed representation can be used in classification or recognition tasks.

Assume that the sparse codes of the T training vectors $\{\mathbf{x}_i\}_{i=1}^T$ are given by $\{\mathbf{a}_i\}_{i=1}^T$ and the indices of the training vectors belonging to class p are given by $\mathcal{C}_p$. Each training vector belongs to one of the P classes. Denoting the mean and variance for each class as $\mathbf{m}_i$ and σ_i^2 respectively, they can be computed the sample mean and variance estimates of coefficient vectors corresponding to a class. The Fisher's discriminant is defined as

$$F(\mathbf{A}) = \frac{\| \sum_{p=1}^{P} T_p(\mathbf{m}_p - \mathbf{m})(\mathbf{m}_p - \mathbf{m})^T \|_F^2}{\sum_{p=1}^{P} \sigma_p^2}, \tag{2.37}$$

where $\mathbf{m}$ is the sample mean estimate of all coefficient vectors and T_p is the number of training vectors that belong to class p. Incorporating the Fisher's discriminant, the objective function that focuses only on discrimination can be written as

$$\underset{\mathbf{A}}{\operatorname{argmax}}\, F(\mathbf{A}) - \lambda \sum_{i=1}^{T} \|\mathbf{a}_i\|_0, \tag{2.38}$$

whereas the objective function that combines both representation on a dictionary $\mathbf{D}$ and discrimination can be posed as

$$\underset{\mathbf{A}}{\operatorname{argmax}}\, F(\mathbf{A}) - \lambda_1 \sum_{i=1}^{T} \|\mathbf{x}_i - \mathbf{D}\mathbf{a}_i\|_2^2 - \lambda_2 \sum_{i=1}^{T} \|\mathbf{a}_i\|_0. \tag{2.39}$$

This was one of the first models that included explicit discrimination constraints in sparse representation based classification [93]. Models that perform supervised dictionary learning using supervised sparse coding [94], and by directly incorporating the SVM model parameters [95] have also been proposed.

2.6.2 Sparse Coding based Subspace Identification

The sparse representation of signals, though not developed for classification tasks, can be discriminative since it selects the most compact set of dictionary elements. Sparse representations can be effectively used for classification, if the overcomplete dictionary is constructed using the training samples itself [96], [97]. If a sufficient number of samples are available from each class, it is possible to represent a test sample using a small subset of training samples from the same class. Loosely speaking, by computing the sparsest representation automatically discriminates between the different classes. It can be clearly observed that this approach is a generalization of the nearest neighbor and the nearest subspace algorithms.

The important challenge in performing feature-based recognition of objects and faces is in deciding a suitable low-dimensional image level feature that is informative and still discriminative between classes. Several approaches such as the Eigenfaces, the Fisherfaces and the Laplacianfaces have been very successfully used to efficiently project high-dimensional data to low-dimensional feature spaces. However, the theory of compressed sensing suggests that even random projections can be employed and hence the choice of the suitable feature space in no longer critical in sparse representation frameworks. Formally stating the problem, we are provided with a set of face images from k different classes and the images are vectorized and stacked as columns in the input matrix $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_P]$. Here, the matrix $\mathbf{X}_p = [\mathbf{x}_{p,1}, \mathbf{x}_{p,2}, ..., \mathbf{x}_{p,T_p}]$ contains the vectorized images from class p.

Several models have been proposed for exploiting the structure of $\mathbf{X}$ for recognition. An effective model for face recognition is to model samples from a single class to be lying on a linear subspace. Face images with varying lighting and expression have been shown to lie in a low-dimensional subspace, referred to as the *face subspace*. Given a sufficient number of images in each class, a test sample $\mathbf{z}$ from class p will approximately lie in the linear span of the training samples from the same class.

$$\mathbf{z} = \sum_{i=1}^{T_p} \mathbf{a}_{p,i}\mathbf{x}_{p,i}. \tag{2.40}$$

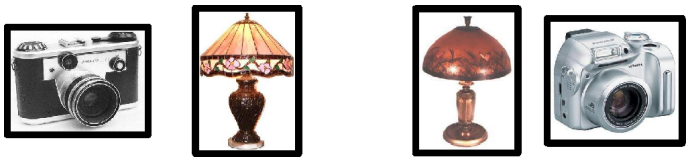
However, the underlying class of the test sample is unknown to begin with and hence we evaluate a linear representation using all images in $\mathbf{X}$. The approximation can be obtained by picking the nearest neighbors to the test sample and computing the coefficients by solving a least squares problem. However, it has been found in [96] that using sparse coding is superior to using nearest neighbor based approaches. This is because, when there are a large number of samples in each class the resulting coefficient vector is naturally sparse. In fact, more sparse the coefficient vector $\mathbf{a}$, easier it is to identify the class membership of $\mathbf{z}$. For the test sample, we compute the coefficient vector and evaluate the residual error with respect to every class. For the p^{th} class we compute the residual error as

$$\mathbf{r}_p(\mathbf{z}) = \|\mathbf{z} - \mathbf{X}\delta_p(\mathbf{a})\|, \tag{2.41}$$

where $\delta_p(\mathbf{a})$ is a new vector with only non-zero coefficients from $\mathbf{a}$ corresponding to the class p. Finally, the test sample is assigned to the class with the least residual error.

2.6.3 Using Unlabeled Data in Supervised Learning

Supervised classification in machine learning often require labeled data that are expensive and difficult to obtain in practice. However, the abundance of unlabelled data motivates design of frameworks that can exploit this information to perform



(a) Semi-supervised learning.

(b) Self-taught transfer learning.

Figure 2.8: Machine learning formalisms to include unlabeled data in supervised learning. The supervised task is to classify lamps and cameras and the self-taught learning framework uses randomly chosen unlabeled data.

supervised tasks. Self-taught transfer learning is a recent machine learning formalism based on sparse coding that provides a principled approach to use unlabeled data. Though semi-supervised learning allows unlabeled data, it makes an additional assumption that the labels of the data are just unobserved and can be labled with the same labels as the supervised task. Figure 2.8 illustrates the two formalisms to include unlabeled data in supervised classification.

Self-taught learning is motivated by the observation that local regions of natural images demonstrate redundancy. In other words, even random images downloaded from the Internet will contain elementary features that are similar to those in the images we want to classify. To formally state the problem, we are provided with a labeled set of training examples $\{(\mathbf{x}_l^1, y^1), (\mathbf{x}_l^2, y^2), (\mathbf{x}_l^3, y^3), ..., , (\mathbf{x}_l^{T_l}, y^{T_l})\}$, where $\mathbf{x}_l^i$ denotes the i^{th} training example and y^i is its corresponding label. In addition, we are provided with a set of unlabeled examples, $\mathbf{x}_u^1, \mathbf{x}_u^2, ..., \mathbf{x}_u^{T_u}$. No assumptions are made regarding the underlying distribution of the unlabeled examples. However, it is assumed that the examples are of the same type as the labeled examples, Eg. images, audio. The algorithm proposed in [98] uses the raw pixel intensities in $\{\mathbf{x}_u^i\}_{i=1}^{T_u}$ to learn the elementary features that comprise an image. As a result, it learns to repre-

sent images in terms of the features rather than in terms of the raw pixel intensities. These learned features are used to obtain an abstract or higher level representation for the labeled data $\{\mathbf{x}_l^i\}_{i=1}^{T_l}$ as well.

The representative features are learned using a modified version of the sparse coding algorithm proposed by Olshausen and Field, which modeled the processing in the primary visual cortex of the humans. Given the set of unlabeled data,

$$\min_{\mathbf{D},\mathbf{a}} \sum_i \|\mathbf{x}_u^i - \sum_j a_j^i \mathbf{d}_j\|_2^2 + \beta\|\mathbf{a}^i\|_1 \qquad \text{subj. to} \qquad \|\mathbf{d}_j\|_2 \leq 1, \forall j. \tag{2.42}$$

In other words, the algorithm learns an overcomplete dictionary using the patches from the unlabeled images. This optimization problem, though not convex jointly, is convex over the subset of variables $\mathbf{D}$ and $\mathbf{a} = \{\mathbf{a}^1, \ldots, \mathbf{a}^{T_u}\}$. The optimization over the coefficients is an ℓ_1 regularized least squares problem, while the optimization over the dictionary is an ℓ_2 constrained least squares problem. This iterative optimization can be performed using the dictionary learning approaches.

In general, it is often easy to obtain large amounts of unlabeled data from the internet that shares important features with the labeled data of interest. Hence, the dictionary learned using the unlabeled can generalize well to represent data in the classification task. For each training input from the classification task, we evaluate sparse codes with $\mathbf{D}$ using error-constrained ℓ_1 minimization. Finally, these features can be used to train standard supervised classifiers such as the SVM.

2.6.4 Generalizing Spatial Pyramids

In complex visual recognition tasks, it is common to extract relevant local or global image descriptors instead of using the raw image pixels directly. Scene understanding is typically carried out by building a Bag of Words (BoW) model that collectively represents the set of local features in an image. However, the spatial information about the features is ignored when considering the orderless BoW model. The spatial information can also be exploited by aggregating the features (using histograms)

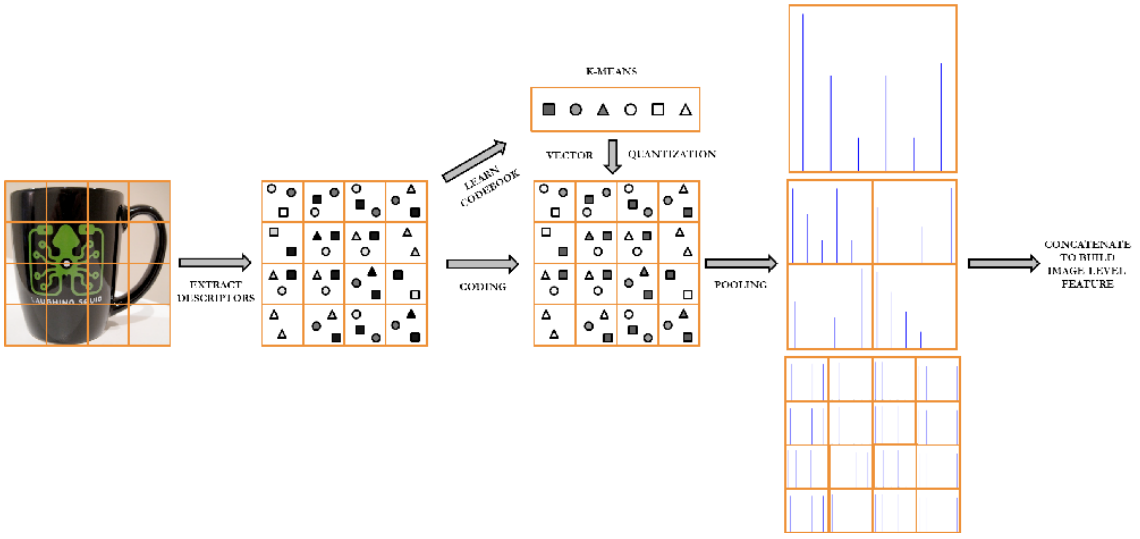


Figure 2.9: Illustration of the process of constructing the spatial pyramid feature for an image. Local descriptors obtained from the patches in the image are used to learn the codebook. Using the codebook, we perform vector quantization of the local descriptors to exploit the redundancy. Finally, we construct the bag of features model at multiple spatial scales. The resulting spatial pyramid feature is the concatenation of histograms from all scales.

at multiple spatial scales, and this process is referred to as constructing a spatial pyramid. Several state-of-the-art object recognition systems involve the construction of spatial pyramid features to achieve improved recognition performance.

The algorithm proposed in [99] partitions the image into increasingly finer regions and evaluates features in the local regions. Typically, local features such as the Scale Invariant Feature Transform (SIFT) or the Histogram of Oriented Gradients (HOG) descriptors are evaluated for small image patches, and the descriptors are coded using vector quantization. Since the local features demonstrate redundancy across multiple images, the descriptors can be efficiently coded using codebooks generated by standard procedures such as the K-means clustering. In a given spatial scale, all code vectors in each sub-region can be aggregated by building histograms. As we move from a coarser to a finer spatial scale, the number of sub-regions increase. The aggregated feature provides a trade-off between translation invariance

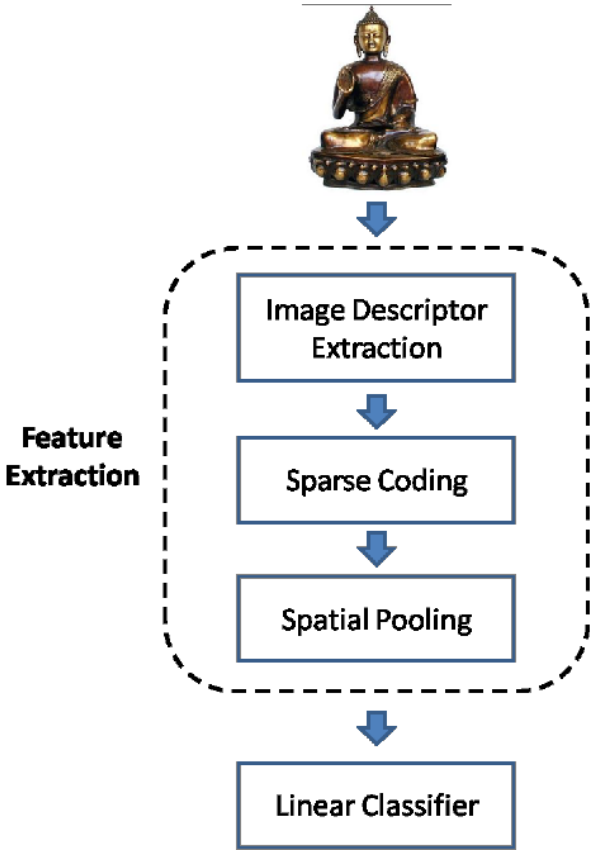


Figure 2.10: Demonstration of linear spatial pyramid matching based on sparse coding. Note that the vector quantization step in the conventional non-linear SPM is replaced by sparse coding, and the underlying spatial pooling process is *max pooling*.

and spatial locality. In the coarsest spatial scale where the whole image is considered, the max pooling feature achieves translation invariance of the local patterns. On the other hand, as we proceed towards finer spatial scales, spatial information is efficiently captured. By concatenating the set of histograms obtained at all spatial scales, we can construct the spatial pyramid. Figure 2.9 illustrates the steps involved in the spatial pyramid matching algorithm.

The spatial pyramid feature, when combined with a non-linear classifier, has been shown to achieve high recognition accuracies. Though, this approach has been very effective, the complexity of using a non-linear classifier is quite high. Using linear kernels with histogram features often leads to substantially worse results, partially

due to the high quantization error in vector quantization. However, when using non-linear kernels, the computational cost of computing the kernel ($\mathcal{O}(T^3)$) and storage requirements ($\mathcal{O}(T^2)$) are quite high, particular for large values of T (total number of training features) [100]. Furthermore, as the number of support vectors scales to the number of training samples, testing cost also increases. As a result, there is a need to employ a different feature extraction mechanism such that the classification can be efficiently performed using only linear classifiers.

The ScSPM algorithm proposed in [100] addresses this problem by computing sparse codes of the local image descriptors, and performing non-linear aggregation (or pooling). Given a pre-defined dictionary $\mathbf{D}$ and the set of descriptors $\mathbf{X}$, we can obtain the sparse codes using any algorithm described earlier in this chapter. A suitable image feature can be constructed by applying a pre-defined aggregation function to the coefficient matrix $\mathbf{A}$. In the coefficient matrix, each row corresponds to the response of all descriptors to a particular dictionary atom. Though a number of pooling functions can be used, the *max pooling* function has been found to perform the best. By performing *max pooling* on $\mathbf{A}$, we can obtain obtain the feature vector $\mathbf{z} \in \mathbb{R}^K$ as follows

$$z_k = \max\{|a_{k,1}|, |a_{k,2}|, \ldots, |a_{k,T}|\}, \text{for } k = 1 \text{ to } K. \tag{2.43}$$

Here $|a_{k,i}|$ denotes the element in the k^{th} row, i^{th} column of the coefficient matrix, T and K denote the total number of descriptors in the image and dictionary atoms respectively. The *max pooling* process has been commonly employed in mathematical models for the visual cortex (V1) and empirically justified by many algorithms. Similar to the construction of the spatial pyramid feature, *max pooling* is performed across multiple locations, and different spatial scales of the image. Hence, the aggregated feature is more robust to local transformations when compared to using histograms. Figure 2.10 illustrates the steps involved in the ScSPM algorithm. Though the con-

ventional spatial pyramid feature leads to very poor performances with a linear kernel, linear SPM kernel based on sparse coding statistics can achieve high classification accuracy. This behavior can be attributed to the better approximation power of sparse coding in natural images and the discrimination power of the aggregated spatial pyramid feature. In addition to object recognition, ScSPM features have been used for SAR target recognition [101], and efficient image retrieval [102].

Supervised Dictionary Optimization

In applications such as face recognition, evaluating sparse representation of a test sample in terms of its training examples and allowing some error for occlusion, fails to effectively handle variations such as pose and alignment. As a result, extracting image-level features that are invariant to global misalignment can result in improved recognition performance [103]. As described earlier, the ScSPM feature provides a trade-off between translation invariance and spatial locality. The efficiency of these features can be improved further by optimizing the dictionary $\mathbf{D}$ to obtain ScSPM features with higher discrimination.

For a training image $\mathbf{X}_t$, we can obtain the coefficient matrix $\mathbf{A}_t$ by performing sparse coding using a known dictionary $\mathbf{D}$. In order to construct the ScSPM feature, we perform *max pooling* of the coefficients in all cells of each spatial scale. We denote the set-level max pooled feature in the c^{th} spatial cell of the s^{th} spatial scale by $\mathbf{z}_c^s$. Assuming that there are R spatial scales, in each scale the image is divided into $2^{s-1} \times 2^{s-1}$ non-overlapping cells. The hierarchical ScSPM is feature is constructed by concatenating all set-level features.

$$\mathbf{z}_t = \eta_{max}(\mathbf{A}_t) = \bigcup_{s=1}^{R} \left[\bigcup_{c=1}^{2^{s-1}} \mathbf{z}_c^s \right]. \tag{2.44}$$

For classification, let us assume a predictive model $f(\mathbf{z}_t, \mathbf{w})$, a loss function $\ell(y_t, f(\mathbf{z}_t, \mathbf{w}))$, where y_k denotes the class label and $\mathbf{w}$ indicates the predictive model parameters. The objective function for obtaining the ScSPM features, dictionary, and

the predictive model parameters can be expressed as

$$E(\mathbf{D}, \mathbf{w}, \{\mathbf{X}_t\}_{t=1}^T) = \sum_{t=1}^{T} \ell(y_t, f(\mathbf{z}_t, \mathbf{w})) + \lambda \|\mathbf{w}\|_2^2. \tag{2.45}$$

This optimization can be performed alternatively with respect to $\mathbf{D}$ and $\mathbf{w}$. Give the dictionary, solving for $\mathbf{w}$ is equivalent to training a classifier, and hence any standard algorithm can be used. However, in order to optimize E over $\mathbf{D}$, we compute the gradient as

$$\frac{\partial E}{\partial \mathbf{D}} = \sum_{t=1}^{T} \frac{\partial \ell}{\partial \mathbf{D}} = \sum_{t=1}^{T} \frac{\partial \ell}{\partial f} \frac{\partial f}{\partial \mathbf{z}_t} \frac{\partial \mathbf{z}_t}{\partial \mathbf{A}_t} \frac{\partial \mathbf{A}_t}{\partial \mathbf{D}}. \tag{2.46}$$

This gradient is evaluated using implicit differentiation [103], and finally the dictionary is updated iteratively such that the different classes are separable.

Chapter 3

CHARACTERISTICS AND IMPLEMENTATION OF 1-D SUBSPACE CLUSTERING

The K-lines clustering algorithm is an iterative procedure that performs a least squares fit of K 1-D subspaces to the training data [23, 35]. When contrasted with K-means clustering, it allows each data sample to have an arbitrary coefficient value corresponding to the associated cluster centroid and the cluster centroids are normalized to unit ℓ_2 norm.

3.1 The K-lines Clustering Algorithm

Let us assume that the data $\mathcal{Y}$ lies in $\mathbb{R}^M$ and define the probability space $(\mathcal{Y}, \mathbf{\Sigma}, P)$, where P is an unknown probability measure. The training samples, $\{\mathbf{y}_i\}_{i=1}^T$, are T i.i.d. realizations from the probability space. We also define an empirical probability measure P_T that assigns the mass T^{-1} to each of the T training samples [104]. The goal of the clustering is to find K partitions of the training data that results in minimum total distortion.

The K-lines clustering algorithm is similar to the K-means algorithm and it is an alternating minimization problem that proceeds in two stages after initialization: the cluster assignment and the cluster centroid update stages. In the cluster assignment stage, the training vector $\mathbf{y}_i$ is assigned to a cluster j based on the minimum distortion criteria, $\mathcal{H}(\mathbf{y}_i) = \text{argmin}_j\, d(\mathbf{y}_i, \boldsymbol{\psi}_j)$, which is equivalent to $\mathcal{H}(\mathbf{y}_i) = \text{argmax}_j\, |\mathbf{y}_i^T \boldsymbol{\psi}_j|$. Here, $\mathcal{H}(.)$ is the membership function that returns the cluster index of a training vector and the distortion measure is

$$d(\mathbf{y}, \boldsymbol{\psi}) = \|\mathbf{y} - \boldsymbol{\psi}(\mathbf{y}^T \boldsymbol{\psi})\|_2^2, \tag{3.1}$$

where $\boldsymbol{\psi}$ is assumed to be normalized. The set $\mathcal{C}_j = \{i | \mathcal{H}(\mathbf{y}_i) = j\}$ contains training vector indices corresponding to the cluster j. Ties in the cluster assignment stage are

broken arbitrarily. Based on the cluster assignment, the updated cluster centroids can be obtained as described in Section 3.1.1.

3.1.1 Computation of Cluster Centroids

Given the set $\mathcal{C}_j$, the j^{th} cluster centroid is updated as $\boldsymbol{\psi}_j = \text{argmin}_{\boldsymbol{\psi}}$ $\mathbf{E}_{P_T}[d(\mathbf{y}, \boldsymbol{\psi})|\mathcal{H}(\mathbf{y}) = j]$. This can also be expressed using the following equation.

$$\boldsymbol{\psi}_j = \underset{\boldsymbol{\psi}}{\text{argmin}} \sum_{i \in \mathcal{C}_j} \|\mathbf{y}_i - \boldsymbol{\psi}(\mathbf{y}_i^T \boldsymbol{\psi})\|_2^2. \tag{3.2}$$

Consider the matrix $\mathbf{Y}_j = [\mathbf{y}_i]_{i \in \mathcal{C}_j}$ and the Singular Value Decomposition (SVD) of $\mathbf{Y}_j = \mathbf{U}_j \boldsymbol{\Upsilon}_j \mathbf{V}_j^T$, where $\boldsymbol{\Upsilon}_j$ is a diagonal matrix with singular values arranged in descending order. The solution to (3.2) is obtained by taking $\boldsymbol{\psi}_j$ as the first column of $\mathbf{U}_j$. Note that the K-lines clustering is not a Bregman divergence based clustering scheme since the centroid is not the conditional expectation of the training vectors [105].

Let us assume a generative model for the training vectors $\{\mathbf{y}_i\}_{i \in \mathcal{C}_j}$ in cluster j as, $\mathbf{y}_i = a_{ji} \boldsymbol{\psi}_j + \mathbf{n}_i$, where $\mathbf{n}_i$ are i.i.d. realizations from $\mathcal{N}(0, \sigma^2 \mathbf{I})$ and a_{ji} are arbitrary coefficients. For this model, the Maximum Likelihood (ML) estimate of the cluster centroid $\boldsymbol{\psi}_j$ is obtained using SVD as described earlier and $a_{ji} = \mathbf{y}_i^T \boldsymbol{\psi}_j$. If we constrain $a_{ji} = 1$, the ML estimate of $\boldsymbol{\psi}_j$ is the mean of $\{\mathbf{y}_i\}_{i \in \mathcal{C}_j}$, which is similar to the case of K-means clustering. This illustrates an important difference between K-means and K-lines clustering, as the latter is invariant to arbitrary scaling of the underlying representative pattern. This also motivates the use of SVD based learning algorithms such as the ones proposed in [83] and [106], for sparse representations.

3.1.2 Distortion Function for Clustering

Mathematically, the K-lines clustering is a problem of finding normalized centroids that minimize the total distortion,

$$\begin{aligned} D(\mathcal{H}) &= \frac{1}{T}\sum_{j=1}^{K}\sum_{i\in\mathcal{C}_j} d(\mathbf{y}_i, \boldsymbol{\psi}_j), \\ &= \frac{1}{T}\sum_{j=1}^{K}\sum_{i\in\mathcal{C}_j} \mathbf{y}_i^T\left(\mathbf{I}-\boldsymbol{\psi}_j\boldsymbol{\psi}_j^T\right)\mathbf{y}_i. \end{aligned} \tag{3.3}$$

Note that since $\|\boldsymbol{\psi}_j\|_2 = 1$, $\boldsymbol{\psi}_j\boldsymbol{\psi}_j^T$ has only one non-zero eigen value which is 1. Hence $\left(\mathbf{I}-\boldsymbol{\psi}_j\boldsymbol{\psi}_j^T\right)$ is positive semi-definite and $\mathbf{y}_i^T\left(\mathbf{I}-\boldsymbol{\psi}_j\boldsymbol{\psi}_j^T\right)\mathbf{y}_i$ is a positive semidefinite quadratic form. Therefore, $d(\mathbf{y},\boldsymbol{\psi})$ is convex in $\mathbf{y}$. If we denote $\angle(.,.)$ as the smallest angle between two vectors and $\|\boldsymbol{\psi}\|_2 = 1$, then

$$d(\mathbf{y},\boldsymbol{\psi}) = \|\mathbf{y}\|_2^2 \sin^2(\angle(\mathbf{y},\boldsymbol{\psi})). \tag{3.4}$$

K-lines clustering can be posed as a problem of empirical risk minimization over the distortion function class [107]

$$\mathcal{G}_K = \left\{ g_{\boldsymbol{\Psi}}(\mathbf{y}) = d(\mathbf{y},\boldsymbol{\psi}_j), j = \underset{l\in\{1,\cdots,K\}}{\operatorname{argmax}} |\mathbf{y}^T\boldsymbol{\psi}_l| \right\}, \tag{3.5}$$

where $\boldsymbol{\Psi} = \{\boldsymbol{\psi}_j\}_{j=1}^K$. The class $\mathcal{G}_K$ is obtained by taking functions corresponding to all possible combinations of K lines from the $\mathbb{R}^M$ space for the set $\boldsymbol{\Psi}$. Minimizing the distortion for the training samples $\{\mathbf{y}_i\}_{i=1}^T$ is equivalent to finding

$$g_{\hat{\boldsymbol{\Psi}}} = \underset{g\in\mathcal{G}_K}{\operatorname{argmin}} \frac{1}{T}\sum_{i=1}^{T} g_{\boldsymbol{\Psi}}(\mathbf{y}_i). \tag{3.6}$$

This is the same as finding the function g corresponding to minimum expected risk of learning (i.e.), $g_{\hat{\boldsymbol{\Psi}}} = \operatorname{argmin}_{g\in\mathcal{G}_K} \mathbf{E}_{P_T}[g_{\boldsymbol{\Psi}}]$, for the probability measure P_T [108].

3.1.3 Covering Number for the Distortion Function Class

Similar to [26], we will show that the covering number for the function class $\mathcal{G}_K$ with respect to the supremum norm, $N_\infty(\epsilon,\mathcal{G}_K)$, is polynomial.

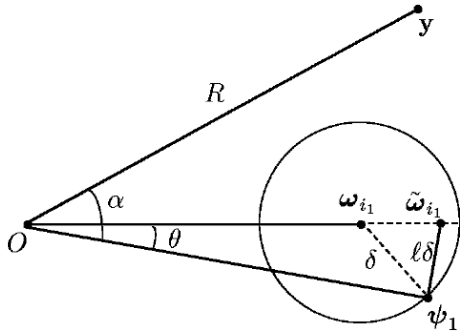


Figure 3.1: Illustration of the geometry for finding the covering number of $\mathcal{G}_K$ with respect to the supremum norm.

Lemma 3.1.1 *For the data lying in an M-dimensional ℓ_2 ball of radius R centered at the origin, the covering number for the distortion function class $\mathcal{G}_K$, with respect to the supremum norm, is upper bounded for $\epsilon > 0$ as*

$$N_\infty(\epsilon, \mathcal{G}_K) \leq \left(\frac{8R^3K + \epsilon}{\epsilon} \right)^{MK}.$$

Proof Let us assume that the radius of the ℓ_2 ball $R > 1$, without loss of generality. This is required to cover the cluster centroids ($\boldsymbol{\Psi} = \{\boldsymbol{\psi}_j\}_{j=1}^K$) which are normalized to lie on the surface of the unit ball. In case of $R \leq 1$, we can renormalize the cluster centroids to a norm less than R and the clustering performance will be unaffected.

Let Q be the number of ℓ_2 balls with radius δ that cover the ball of radius R. We know that

$$Q = \left(\frac{4R + \delta}{\delta} \right)^M, \tag{3.7}$$

in M dimensions [104]. The centers of the $\delta-$balls are given by the set $\Omega = \{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, ..., \boldsymbol{\omega}_Q\}$. To find the upper bound on $N_\infty(\epsilon, \mathcal{G}_K)$, we will successively move each cluster centroid $\boldsymbol{\psi}_j$ to the center of the nearest $\delta-$ball denoted by $\boldsymbol{\omega}_{i_j}$, such that $\|\boldsymbol{\psi}_j - \boldsymbol{\omega}_{i_j}\|_2 \leq \delta$. The cluster centroid that replaces $\boldsymbol{\psi}_j$ is given by $\tilde{\boldsymbol{\omega}}_{i_j} = \boldsymbol{\omega}_{i_j}/\|\boldsymbol{\omega}_{i_j}\|_2$.

For the case of $j = 1$, let us assume $\angle(\tilde{\boldsymbol{\omega}}_{i_1}, \boldsymbol{\psi}_1) = \theta$ and $\angle(\mathbf{y}, \boldsymbol{\psi}_1) = \alpha$, where $\mathbf{y}$ is any data vector. This setup is illustrated in Figure 3.1. Assuming that

$\|\boldsymbol{\psi}_j - \tilde{\boldsymbol{\omega}}_{i_j}\|_2 \leq l\delta$, and using the fact that any line segment contained in a $\delta-$ball has a length less than 2δ, we have $0 \leq l \leq 2$. The updated set of centroids for $j = 1$ is given by $\boldsymbol{\Psi}_1 = \{\tilde{\boldsymbol{\omega}}_{i_1}, \boldsymbol{\psi}_2, \cdots, \boldsymbol{\psi}_K\}$ and this is obtained by replacing $\boldsymbol{\psi}_1$ with $\tilde{\boldsymbol{\omega}}_{i_1}$.

The supremum of $|g_{\boldsymbol{\Psi}} - g_{\boldsymbol{\Psi}_1}|$ can occur only when $\|\mathbf{y}\|_2 = R$ and when $\mathbf{y}$, $\boldsymbol{\psi}_1$ and $\tilde{\boldsymbol{\omega}}_{i_1}$ are co-planar. For a particular θ, using (3.4), the supremum can be bounded as

$$\begin{aligned} \|g_{\boldsymbol{\Psi}} - g_{\boldsymbol{\Psi}_1}\|_\infty &\leq \max_{\alpha} \left[R^2 \sin^2 \alpha - R^2 \sin^2(\alpha - \theta)\right], \\ &= R^2 \sin\theta, \end{aligned} \tag{3.8}$$

and this is obtained when $\alpha = \pi/4 + \theta/2$. We also know from Figure 3.1 that $\sin\theta/2 = l\delta/2$ and hence,

$$\begin{aligned} \|g_{\boldsymbol{\Psi}} - g_{\boldsymbol{\Psi}_1}\|_\infty &\leq R^2 l\delta\sqrt{(1 - l^2\delta^2/4)}, \\ &\leq R^2 l\delta. \end{aligned} \tag{3.9}$$

Performing this operation for all the K centroids, we obtain,

$$\|g_{\boldsymbol{\Psi}} - g_{\boldsymbol{\Psi}_K}\|_\infty \leq R^2 Kl\delta. \tag{3.10}$$

From (3.10), we obtain $\epsilon = R^2 Kl\delta$ and hence $\delta = \epsilon/KlR^2$. There are at most Q^K combinations of cluster centroids that cover $g_{\boldsymbol{\Psi}}$ within an L_∞ distance, ϵ. Hence the upper bound on $N_\infty(\epsilon, \mathcal{G}_K)$ is obtained by substituting for δ in Q^K and replacing l with its upper bound value.

It can be observed from Lemma 3.1.1 that the covering number grows polynomially. Hence, the class of functions $\mathcal{G}_K$ is uniform Donsker [26]. Being uniform Donsker, the empirical average of functions $g \in \mathcal{G}_K$ uniformly converge to their expected value $\mathbf{E}_P[g]$ as $T \to \infty$, with respect to the measure P [108].

3.1.4 Voronoi Tessellation by K lines

The Voronoi regions obtained from a set of points in $\mathbb{R}^M$ with respect to the Euclidean distance measure are convex, whereas a Voronoi construction obtained from a set of lines is not convex in general [109]. However, we will show that for any set of K lines, convex Voronoi regions are obtained with respect to the distortion measure defined in (3.1). This implies that the K centroids tessellate the space into convex Voronoi regions.

Lemma 3.1.2 *Let $\{\boldsymbol{\psi}_j\}_{j=1}^K$ represent a set of K lines in the $\mathbb{R}^M$ space. With respect to the distortion measure defined in (3.1), the set of K lines tessellate the $\mathbb{R}^M$ space into $2K$ convex Voronoi regions.*

Proof To show this, we prove that the K lines form $2K$ convex polytopes that tessellate the $\mathbb{R}^M$ space. Assume that the lines are of unit norm and they are indexed by the set $\mathcal{K} = \{1, 2, ..., K\}$, using the index variable $j \in \mathcal{K}$. The index of the closest line to each vector $\mathbf{y} \in \mathbb{R}^M$ is given by $\mathcal{H}(\mathbf{y}) = \text{argmax}_j \, |\mathbf{y}^T \boldsymbol{\psi}_j|$. The region for the line $\boldsymbol{\psi}_j$ contains the vectors given by the set $\mathcal{R}_j = \{\mathbf{y} \mid \mathcal{H}(\mathbf{y}) = j\}$. We form an extended set of $2K$ unit vectors $\{+\boldsymbol{\psi}_j, -\boldsymbol{\psi}_j\}_{j=1}^K$ and define $\boldsymbol{\omega}_{2j-1} = +\boldsymbol{\psi}_j, \boldsymbol{\omega}_{2j} = -\boldsymbol{\psi}_j$. The set $\boldsymbol{\Omega} = \{\omega_l\}_{l=1}^{2K}$ and its elements are indexed by the set $\hat{\mathcal{K}} = \{1, 2, ..., 2K\}$, using the index variable $l \in \hat{\mathcal{K}}$. For any vector $\mathbf{y}$ in $\mathbb{R}^M$, we can find the closest vector in $\boldsymbol{\Omega}$ using $\hat{\mathcal{H}}(\mathbf{y}) = \text{argmax}_l \, \langle \mathbf{y}, \boldsymbol{\omega}_l \rangle$. The region for $\boldsymbol{\omega}_l$ contains the vectors given by $\hat{\mathcal{R}}_l = \{\mathbf{y} \mid \hat{\mathcal{H}}(\mathbf{y}) = l\}$. Note that $\hat{\mathcal{R}}_{2j-1}$ contains the negative of the vectors in $\hat{\mathcal{R}}_{2j}$ and $\mathcal{R}_j = \hat{\mathcal{R}}_{2j-1} \cup \hat{\mathcal{R}}_{2j}$, for $j \in \mathcal{K}$. The union of all the regions $\bigcup_{l=1}^{2K} \hat{\mathcal{R}}_l$ contains all the vectors in $\mathbb{R}^M$, if we break the ties when computing $\hat{\mathcal{H}}(\mathbf{y})$ based on the magnitude of the region index l. Hence we prove that the set of K lines tessellate the $\mathbb{R}^M$ space into $2K$ regions.

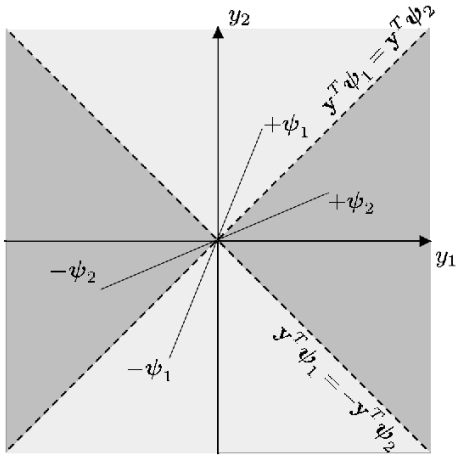


Figure 3.2: Tessellation of the 2-D space by the lines $\boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_2$ into four convex Voronoi regions.

To prove that the regions $\hat{\mathcal{R}}_l$ are convex, we consider the boundary that separates the two adjacent regions $\hat{\mathcal{R}}_l$ and $\hat{\mathcal{R}}_m$, $\mathbf{y}_{l,m} = \{\mathbf{y} \mid \mathbf{y}^T\boldsymbol{\omega}_l = \mathbf{y}^T\boldsymbol{\omega}_m\}$. The boundary is an $M-1$ dimensional subspace that partitions the $\mathbb{R}^M$ space into two halfspaces. Similarly, we can find a boundary subspace for each of the adjacent regions of $\hat{\mathcal{R}}_l$. Therefore the region $\hat{\mathcal{R}}_l$ is a polytope formed by the intersection of halfspaces defined by the boundary subspaces and hence convex. Because of our tie-breaking condition some regions may contain the boundary subspaces as well, but the regions will be still convex.

The Voronoi tessellation of 2 lines, $\boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_2$, of a 2-dimensional space is illustrated in Figure 3.2. The boundary hyperplanes (shown by dotted lines) represented by $\mathbf{y}^T\boldsymbol{\psi}_1 = \mathbf{y}^T\boldsymbol{\psi}_2$ and $\mathbf{y}^T\boldsymbol{\psi}_1 = -\mathbf{y}^T\boldsymbol{\psi}_2$ separate the space into 4 convex regions.

3.2 Local Optimality

We show the optimality of the clustering algorithm using the Lloyd's optimality conditions. The optimality is also ensured by showing that K-lines clustering is a special case of the general dictionary learning problem for sparse representations

using an EM formulation. The K-lines clustering algorithm is optimal with respect to the probability measure P_T defined in Section 3.1. This can also be shown using the Lloyd's optimality conditions [110], as learning the cluster centroids is similar to learning a shape-gain vector quantizer with unquantized gain values. Note that the Lloyd's conditions used here are the conditions for optimality of a shape quantizer. However, we can use it for our clustering scheme as we do not quantize the gains and given the shape $\boldsymbol{\psi}$, the gain $a = \mathbf{y}^T\boldsymbol{\psi}$ is uniquely defined. Any vector quantizer is locally optimal, if it satisfies the following conditions:

1. *Nearest Neighbor Condition:* The clustering satisfies the nearest neighbor condition, as each vector in the $\mathbb{R}^M$ space is assigned to the nearest cluster centroid using the distance metric given in (3.1).

2. *Centroid Condition:* The T training vectors lie in a probability space with the probability measure P_T. The centroid update stage in the clustering algorithm computes the centroid $\boldsymbol{\psi}_j = \text{argmin}_{\boldsymbol{\psi}}\, E_{P_T}[d(\mathbf{y}, \boldsymbol{\psi})|\mathcal{H}(\mathbf{y}) = j]$, as given in Section 3.1.1. This satisfies one of the sufficient conditions for optimality.

3. *Zero Probability on Boundary Condition:* This condition is satisfied for the K-lines clustering algorithm with a tie-breaking scheme. In this case, the algorithm never assigns any training vector to more than one cluster centroid and hence the probability of finding a training vector in the cluster boundary is always zero.

Since the K-lines algorithm satisfies the sufficient conditions for optimality, it is locally optimal [110].

3.3 Stability

It was shown in Section 3.1.3 that the distortion function of K-lines clustering belongs to the uniform Donsker class. Therefore, assuming the existence of a unique

minimizer, the distortion functions corresponding to two different clusterings $\mathbf{\Psi}$ and $\mathbf{\Lambda}$ become arbitrarily close, $\|g_{\mathbf{\Psi}} - g_{\mathbf{\Lambda}}\|_{L_1(P)} \xrightarrow{P} 0$, as the number of training samples $T \to \infty$. Here we assume that the clusterings are trained with different sets of i.i.d. training samples drawn from the probability space $(\mathcal{Y}, \mathbf{\Sigma}, P)$. If a unique minimizer does not exist, then the distortion function is stable with respect to a change in $o(\sqrt{T})$ points [26]. We will follow a line of reasoning similar to [26] in showing that the cluster centroids are also stable, given that the distortion functions are stable.

The distortion function $g_{\mathbf{\Psi}}$ can also be expressed as

$$g_{\mathbf{\Psi}}(\mathbf{y}) = \sum_{j=1}^{K} d(\mathbf{y}, \boldsymbol{\psi}_j)\, \mathbb{I}\left(\mathbf{y} \text{ closest to } \boldsymbol{\psi}_j\right), \tag{3.11}$$

where $\mathbb{I}$ is the indicator function. Let us denote the two cluster centroids as $\mathbf{\Psi} = \{\boldsymbol{\psi}_1, ..., \boldsymbol{\psi}_K\}$ and $\mathbf{\Lambda} = \{\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_K\}$ and define the distance between the cluster centroids [26] as

$$\Delta(\mathbf{\Psi}, \mathbf{\Lambda}) = \max_{1 \le j \le K} \min_{1 \le l \le K} \left[(d(\boldsymbol{\psi}_j, \boldsymbol{\lambda}_l))^{1/2} + (d(\boldsymbol{\psi}_l, \boldsymbol{\lambda}_j))^{1/2} \right]. \tag{3.12}$$

Theorem 3.3.1 *If the L_1 distance between the distortion functions for the clusterings $\mathbf{\Psi}$ and $\mathbf{\Lambda}$ is bounded as $\|g_{\mathbf{\Psi}} - g_{\mathbf{\Lambda}}\|_{L_1(P)} < \epsilon$ and $dP(\mathbf{y})/d\mathbf{y} > C$, then $\Delta(\mathbf{\Psi}, \mathbf{\Lambda}) \le 2\sin(\rho)$ where*

$$\rho \le 2\sin^{-1}\left[\frac{1}{r}\left(\frac{\epsilon}{\hat{C}_{C,M}}\right)^{\frac{1}{M+1}}\right], \tag{3.13}$$

when the training data lies outside an M-dimensional ℓ_2 ball of radius r centered at the origin. Here $\epsilon > 0$ and the constant $\hat{C}_{C,M}$ depends only on C and M.

Proof The distance between the cluster centroids given in (5.14) can be bounded as

$$Lambda\Delta(\mathbf{\Psi}, \mathbf{\Lambda}) \le 2\max\left[\max_{1 \le j \le K} \min_{1 \le l \le K} (d(\boldsymbol{\psi}_j, \boldsymbol{\lambda}_l))^{1/2}, \max_{1 \le j \le K} \min_{1 \le l \le K} (d(\boldsymbol{\psi}_l, \boldsymbol{\lambda}_j))^{1/2}\right].$$

Without loss of generality, we assume that the maximum occurs at $(\boldsymbol{\psi}_1, \boldsymbol{\lambda}_1)$ and hence we have $\Delta(\mathbf{\Psi}, \mathbf{\Lambda}) \le 2(d(\boldsymbol{\psi}_1, \boldsymbol{\lambda}_1))^{1/2}$. Denoting $\angle(\boldsymbol{\psi}_1, \boldsymbol{\lambda}_1) = \rho$, we have

$\Delta(\mathbf{\Psi}, \mathbf{\Lambda}) \leq 2\sin(\rho)$, using (3.4). Let us define a region $S(\boldsymbol{\psi}_1, \rho/2) = \{\mathbf{y} | \angle(\boldsymbol{\psi}_1, \mathbf{y}_1) \leq \rho/2, \|\mathbf{y}\|_2 \geq r\}$. If $\mathbf{y} \in S(\boldsymbol{\psi}_1, \rho/2)$, then $\mathbf{y}^T \left(\mathbf{I} - \boldsymbol{\psi}_1\boldsymbol{\psi}_1^T\right)\mathbf{y} \leq \mathbf{y}^T \left(\mathbf{I} - \boldsymbol{\lambda}_1\boldsymbol{\lambda}_1^T\right)\mathbf{y}$. An illustration of this setup for a two dimensional case is given in Figure 3.3. It can be seen from the figure that $\angle(\mathbf{q}_1, \boldsymbol{\psi}_1) = \angle(\mathbf{q}_2, \boldsymbol{\psi}_1) = \angle(\mathbf{q}_2, \boldsymbol{\lambda}_1) = \rho/2$. The arc $\hat{\mathbf{q}}_1\hat{\mathbf{q}}_2$ with radius r, represents the minimum ℓ_2 norm possible for the data $\mathbf{y}$. We also define the following: $\hat{r} = r\sin(\rho/2)$, $\bar{r} = r\cos(\rho/2)$, $\bar{S}(\boldsymbol{\psi}_1, \rho/2) = \{\mathbf{y} | \angle(\boldsymbol{\psi}_1, \mathbf{y}) \leq \rho/2, \|\mathbf{y}\|_2 = r\}$ (shown as the arc $\hat{\mathbf{q}}_1\hat{\mathbf{q}}_2$ in Figure 3.3) and $\hat{S}(\boldsymbol{\psi}_1, \rho/2) = \{\mathbf{y} | \mathbf{y}^T\boldsymbol{\psi}_1 = \bar{r}, \|\mathbf{y}\|_2 \leq r\}$ (shown as the segment $\hat{\mathbf{q}}_1\hat{\mathbf{q}}_2$). For any $\mathbf{y} \in \mathcal{Y}$,

$$\mathbf{y}^T \left(\mathbf{I} - \boldsymbol{\psi}_1\boldsymbol{\psi}_1^T\right)\mathbf{y} \geq \sum_{j=1}^{K} \mathbf{y}^T \left(\mathbf{I} - \boldsymbol{\psi}_j\boldsymbol{\psi}_j^T\right)\mathbf{y}\, \mathbb{I}\left(\mathbf{y} \text{ closest to } \boldsymbol{\psi}_j\right) \tag{3.14}$$

The distance between the distortion functions is

$$\begin{aligned}
\|g_{\mathbf{\Psi}} - g_{\mathbf{\Lambda}}\|_{L_1(P)} &= \int |g_{\mathbf{\Psi}}(\mathbf{y}) - g_{\mathbf{\Lambda}}(\mathbf{y})| dP(\mathbf{y}), && (3.15)\\
&\geq \int_{S(\boldsymbol{\psi}_1, \rho/2)} |g_{\mathbf{\Psi}}(\mathbf{y}) - g_{\mathbf{\Lambda}}(\mathbf{y})| dP(\mathbf{y}), && (3.16)\\
&= \int_{S(\boldsymbol{\psi}_1, \rho/2)} \Big[\mathbf{y}^T \left(\mathbf{I} - \boldsymbol{\lambda}_1\boldsymbol{\lambda}_1^T\right)\mathbf{y} \\
&\qquad - \sum_{j=1}^{K} \mathbf{y}^T \left(\mathbf{I} - \boldsymbol{\psi}_j\boldsymbol{\psi}_j^T\right)\mathbf{y}\, \mathbb{I}\left(\mathbf{y} \text{ closest to } \boldsymbol{\psi}_j\right)\Big] dP(\mathbf{y}), && (3.17)\\
&\geq \int_{S(\boldsymbol{\psi}_1, \rho/2)} \Big[\mathbf{y}^T \left(\mathbf{I} - \boldsymbol{\lambda}_1\boldsymbol{\lambda}_1^T\right)\mathbf{y} - \mathbf{y}^T \left(\mathbf{I} - \boldsymbol{\psi}_1\boldsymbol{\psi}_1^T\right)\mathbf{y}\Big] dP(\mathbf{y}), && (3.18)\\
&\geq C \int_{\bar{S}(\boldsymbol{\psi}_1, \rho/2)} \Big[\bar{\mathbf{y}}^T \left(\mathbf{I} - \boldsymbol{\lambda}_1\boldsymbol{\lambda}_1^T\right)\bar{\mathbf{y}} - \bar{\mathbf{y}}^T \left(\mathbf{I} - \boldsymbol{\psi}_1\boldsymbol{\psi}_1^T\right)\bar{\mathbf{y}}\Big] d\bar{\mathbf{y}}, && (3.19)\\
&\geq C \int_{\hat{S}(\boldsymbol{\psi}_1, \rho/2)} \Big[\hat{\mathbf{y}}^T \left(\mathbf{I} - \boldsymbol{\lambda}_1\boldsymbol{\lambda}_1^T\right)\hat{\mathbf{y}} - \hat{\mathbf{y}}^T \left(\mathbf{I} - \boldsymbol{\psi}_1\boldsymbol{\psi}_1^T\right)\hat{\mathbf{y}}\Big] d\hat{\mathbf{y}}, && (3.20)\\
&\geq C \int_{\hat{S}(\boldsymbol{\psi}_1, \rho/2)} \Big[r^2 \sin^2 \rho/2 - \hat{\mathbf{y}}^T \left(\mathbf{I} - \boldsymbol{\psi}_1\boldsymbol{\psi}_1^T\right)\hat{\mathbf{y}}\Big] d\hat{\mathbf{y}}. && (3.21)
\end{aligned}$$

Here, $C < dP(\mathbf{y})/d\mathbf{y}$ for $\mathbf{y} \in S(\boldsymbol{\psi}_1, \rho/2)$. (3.19) and (3.20) hold because $\bar{\mathbf{y}} \in \bar{S}(\boldsymbol{\psi}_1, \rho/2)$, $\hat{\mathbf{y}} \in \hat{S}(\boldsymbol{\psi}_1, \rho/2)$ and hence $\|\bar{\mathbf{y}}\|_2 \leq \|\mathbf{y}\|_2$, $\|\hat{\mathbf{y}}\|_2 \leq \|\bar{\mathbf{y}}\|_2$. Assuming $\beta = \angle(\mathbf{y}, \boldsymbol{\psi}_1)$, the distortion $\hat{\mathbf{y}}^T \left(\mathbf{I} - \boldsymbol{\psi}_1\boldsymbol{\psi}_1^T\right)\hat{\mathbf{y}} = \bar{r}^2 \tan^2 \beta$. Denoting $\gamma = \bar{r}\tan\beta$, the differential element for the region $\hat{S}(\boldsymbol{\psi}_1, \rho/2)$ is given by $d\hat{\mathbf{y}} = (M-1)F_{(M-1)}\gamma^{(M-2)}d\gamma$.

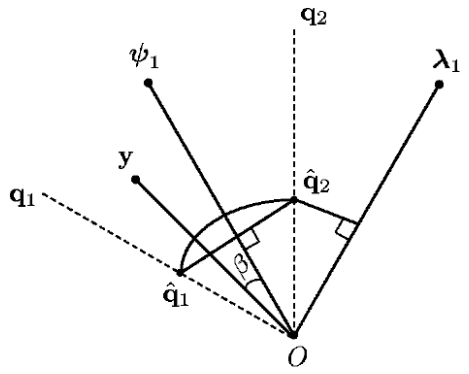


Figure 3.3: Illustration used to show the stability of the cluster centroid from the stability of the distortion function.

This is because $d\hat{\mathbf{y}}$ is the differential volume element of a $(M-1)-$sphere, whose volume is given by $F_{(M-1)}\gamma^{(M-1)}$ for radius γ [111]. Here $F_{(M)} = \pi^{M/2}/\Gamma(M/2+1)$, $\Gamma(.)$ being the gamma function. Now (3.21) can be expressed as

$$C\int_0^{r\sin\rho/2}\left[r^2\sin^2\rho/2-\gamma^2\right](M-1)F_{(M-1)}\gamma^{(M-2)}d\gamma, \tag{3.22}$$

$$=\hat{C}_{C,M}r^{M+1}\sin^{M+1}(\rho/2)\leq\epsilon, \tag{3.23}$$

where $\hat{C}_{C,M} = 2CF_{(M-1)}/(M+1)$. Hence the angle, ρ, between $\boldsymbol{\psi}_1$ and $\boldsymbol{\lambda}_1$ can be bounded as given in (5.15). The bound on angle ρ exists only when $r > (\epsilon/\hat{C})^{1/(M+1)}$.

From Theorem 5.4.1, it can be observed that for admissible values of r, the cluster centroids become arbitrarily close to each other, $\Delta(\boldsymbol{\Psi},\boldsymbol{\Lambda}) \xrightarrow{P} 0$, as $\epsilon \to 0$. This means that the stability of distortion functions implies the stability of cluster centroids. The K-lines clustering cannot be stable if some training vectors have a norm close enough to 0, (i.e.) $r \to 0$. This occurs because the cluster centroids, which are 1-dimensional subspaces, become arbitrarily close to each other near the origin.

3.4 Relationship to Dictionary Learning

We formulate the problem of dictionary learning from a set of training observations using an EM procedure [112]. By placing appropriate constraints, we then show that

the K-lines clustering algorithm is a special case of this process. This also shows that the K-lines clustering algorithm converges to a locally optimal solution. We note here that a similar formulation for learning orthonormal dictionaries has been proposed in [113], although our formulation is more general.

The probabilistic source model for the EM formulation is assumed to be, $\tilde{\mathbf{y}} = \mathbf{\Psi}\tilde{\mathbf{a}} + \tilde{\mathbf{n}}$, and the T independent realizations from the random vector are given by $\mathbf{Y} = [\mathbf{y}_i]_{i=1}^{T}$. The random vector $\tilde{\mathbf{n}} \sim \mathcal{N}(0, \sigma^2\mathbf{I}_M)$, where σ^2 is the noise variance and $\mathbf{I}_M$ is an identity matrix of size $M \times M$. The dictionary is denoted by the matrix $\mathbf{\Psi} = \left[\boldsymbol{\psi}_j\right]_{j=1}^{K}$. The observation matrix $\mathbf{Y}$ and σ^2 are assumed to be known. A sparsity prior is placed on each coefficient vector $\mathbf{a}_i$. The only parameter to be estimated by the EM algorithm is $\mathbf{\Psi}$ and the parameter set $\mathbf{\Theta} = \{\mathbf{\Psi}\}$. The likelihood for $\mathbf{y}_i$ at the iteration l of the EM algorithm is given by

$$p(\mathbf{y}_i|\mathbf{a}_i, \mathbf{\Theta}^{(l)}) = \mathcal{N}(\mathbf{\Psi}^{(l)}\mathbf{a}_i, \sigma^2\mathbf{I}_M). \tag{3.24}$$

The posterior probability of the coefficient vector for the current iteration is computed as

$$p\left(\mathbf{a}_i|\mathbf{y}_i, \mathbf{\Theta}^{(l)}\right) = \frac{p\left(\mathbf{a}_i, \mathbf{y}_i|\mathbf{\Theta}^{(l)}\right)}{p\left(\mathbf{y}_i|\mathbf{\Theta}^{(l)}\right)} = \frac{p\left(\mathbf{a}_i\right) p\left(\mathbf{y}_i|\mathbf{a}_i, \mathbf{\Theta}^{(l)}\right)}{\int_{\mathbf{a}_i} p\left(\mathbf{y}_i, \mathbf{a}_i|\mathbf{\Theta}^{(l)}\right) d\mathbf{a}_i}. \tag{3.25}$$

The expectation that needs to be maximized to find the parameters $\mathbf{\Theta}$ for the next iteration $l+1$ is given by

$$\mathcal{Q}(\mathbf{\Theta}|\mathbf{\Theta}^{(l)}) = \mathbf{E}_{p\left(\{\mathbf{a}_i\}_{i=1}^{T}|\{\mathbf{y}_i\}_{i=1}^{T}, \mathbf{\Theta}^{(l)}\right)}\left[\log p\left(\{\mathbf{y}_i\}_{i=1}^{T}, \{\mathbf{a}_i\}_{i=1}^{T}|\mathbf{\Theta}\right)\right]. \tag{3.26}$$

By independence assumptions we know that

$$\log p\left(\{\mathbf{y}_i\}_{i=1}^{T}, \{\mathbf{a}_i\}_{i=1}^{T}|\mathbf{\Theta}\right) = \sum_{i=1}^{T} \log p\left(\mathbf{y}_i, \mathbf{a}_i|\mathbf{\Theta}\right), \tag{3.27}$$

$$p\left(\{\mathbf{a}_i\}_{i=1}^{T}|\{\mathbf{y}_i\}_{i=1}^{T}, \mathbf{\Theta}^{(l)}\right) = \prod_{i=1}^{T} p\left(\mathbf{a}_i|\mathbf{y}_i, \mathbf{\Theta}^{(l)}\right). \tag{3.28}$$

From (3.26), (3.27) and (3.28) we have

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(l)}) &= \sum_{i=1}^{T}\int_{\mathbf{a}_i}\log p\left(\mathbf{y}_i,\mathbf{a}_i|\boldsymbol{\Theta}\right)p\left(\mathbf{a}_i|\mathbf{y}_i,\boldsymbol{\Theta}^{(l)}\right)d\mathbf{a}_i,\\ &= \sum_{i=1}^{T}\int_{\mathbf{a}_i}\Big[\log\mathcal{N}\left(\mathbf{y}_i|\boldsymbol{\Psi}\mathbf{a}_i,\sigma^2\right)+\log p(\mathbf{a}_i)\Big]p\left(\mathbf{a}_i|\mathbf{y}_i,\boldsymbol{\Theta}^{(l)}\right)d\mathbf{a}_i.\end{aligned}$$

The dictionary is updated in the M-step as

$$\begin{aligned}\boldsymbol{\Psi}^{(l+1)} &= \underset{\boldsymbol{\Psi}}{\operatorname{argmax}}\,\mathcal{Q}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(l)}),\\ &= \underset{\boldsymbol{\Psi}}{\operatorname{argmin}}\sum_{i=1}^{T}\int_{\mathbf{a}_i}\Big[\|\mathbf{y}_i-\boldsymbol{\Psi}\mathbf{a}_i\|_2^2-\log p(\mathbf{a}_i)\Big]p\left(\mathbf{a}_i|\mathbf{y}_i,\boldsymbol{\Theta}^{(l)}\right)d\mathbf{a}_i.\end{aligned}$$

This is the general solution to a dictionary learning problem. In order to obtain the K-lines learning as a special case of this dictionary learning problem, we constrain the posterior density $p(\mathbf{a}_i|\mathbf{y}_i,\boldsymbol{\Theta}^{(l)})$, such that the coefficient vector is 1-sparse. Assuming that $p(\mathbf{a}_i)$ is the same for all i and noting that

$$p\left(\mathbf{a}_i|\mathbf{y}_i,\boldsymbol{\Theta}^{(l)}\right)\propto p\left(\mathbf{y}_i|\mathbf{a}_i,\boldsymbol{\Theta}^{(l)}\right), \tag{3.29}$$

we obtain best the 1-sparse vector by finding

$$\hat{\mathbf{a}}_i = \underset{\mathbf{a}_i}{\operatorname{argmax}}\,p\left(\mathbf{y}_i|\mathbf{a}_i,\boldsymbol{\Theta}^{(l)}\right)\quad\text{s.t.}\quad\|\mathbf{a}_i\|_0=1. \tag{3.30}$$

Using $\hat{\mathbf{a}}_i$, the posterior in (3.25) is constrained as

$$p\left(\mathbf{a}_i|\mathbf{y}_i,\boldsymbol{\Theta}^{(l)}\right)=\begin{cases}1, & \text{if } \mathbf{a}_i=\hat{\mathbf{a}}_i,\\ 0, & \text{if } \mathbf{a}_i\neq\hat{\mathbf{a}}_i.\end{cases} \tag{3.31}$$

We define the membership sets $\mathcal{C}_j^{(l)}=\{i|\hat{a}_{ji}\neq 0\}$ for each dictionary atom $\boldsymbol{\psi}_j$, where $\hat{a}_{ji}$ is the j^{th} element of the vector $\hat{\mathbf{a}}_i$. The dictionary can be updated in the M-step as follows.

$$\boldsymbol{\Psi}^{(l+1)} = \underset{\boldsymbol{\Psi}}{\operatorname{argmin}}\sum_{i=1}^{T}\left(\|\mathbf{y}_i-\boldsymbol{\Psi}\hat{\mathbf{a}}_i\|_2^2-\log p(\hat{\mathbf{a}}_i)\right). \tag{3.32}$$

This can be decoupled for learning each cluster centroid (dictionary atom) as

$$\boldsymbol{\psi}_j^{(l+1)} = \underset{\boldsymbol{\psi}_j}{\operatorname{argmin}} \sum_{i \in \mathcal{C}_j^{(l)}} \|\mathbf{y}_i - \boldsymbol{\psi}_j \hat{a}_{ji}\|_2^2, \tag{3.33}$$

and this equation represents the centroid update stage in the K-lines clustering problem. Because the EM algorithm always converges to a locally optimal solution, K-lines clustering is also locally optimal.

3.5 Efficient Implementation using Random Projections

The very high spatial and temporal dimensionality of the video data makes l_2 norm based clustering intractable in the absence of tremendous computational power. Therefore, it becomes essential to reduce the dimensionality of the video data in order to perform clustering with low complexity. This problem is highly significant in scenarios where fast summarization of video needs to be performed at a reduced computational cost. In this chapter, we propose a framework for dimensionality reduction to cluster video frames having similar background structure. The framework is based on non-adaptive dimensionality reduction using the theory of random projections [114] and assumption of Gaussian mixture (GM) models for data.

3.5.1 Random Projections

Consider a high dimensional data matrix, $\mathbf{X}$, with dimensions $M \times T$, where each column represents a single data observation. In order to project this onto a random low dimensional space, we define a matrix $\mathbf{R}$ of dimensions $M \times N$ with $N < M$, whose entries are chosen independently from the standard normal distribution $\mathcal{N}(0, 1)$. The Random Projection (RP) of the data vectors is,

$$\mathbf{Y} = \frac{1}{\sqrt{N}} \mathbf{R}^T \mathbf{X}. \tag{3.34}$$

RP reduces the dimensionality of the data from M to N while approximately preserving pair-wise distances with high probability [114]. This is formalized by the *Johnson-Lindenstrauss* (JL) lemma, which states that for a large enough N ($N \geq C\frac{\ln T}{\epsilon^2}$) (3.35)

holds with high probability.

$$(1-\epsilon)\left\|\mathbf{y}_i\right\|^2 \leq \frac{1}{N}\|\mathbf{R}^T\mathbf{x}_i\|^2 \leq (1+\epsilon)\left\|\mathbf{y}_i\right\|^2, \tag{3.35}$$

where $\mathbf{y}_i$ and $\mathbf{x}_i$ represent the columns of the matrices $\mathbf{Y}$ and $\mathbf{X}$ respectively and $0 < \epsilon < 1$. It is important to note that the JL lemma does not depend on the actual dimensionality of the data and depends only on the number of data vectors. The K-means clustering defines the cluster centroid as the mean of data vectors in a cluster. It is easy to observe from JL lemma that the distances between the cluster centroid and the data vectors will be approximately preserved even after random projections. Hence we can use the JL lemma to reduce the dimensionality of the data matrix for use in K-means clustering.

3.5.2 Computation of SVD using Random Projections

The computation of SVD can also be performed using the reduced dimensional matrix from random projections [114]. It can be shown that for the same low rank approximations of $\mathbf{Y}$ and $\mathbf{X}$, the Frobenius norms will be approximately preserved with high probability. This can be mathematically expressed as,

$$\sum_{i=1}^{s}\lambda_i^2 \geq (1-\epsilon)\sum_{i=1}^{s}\sigma_i^2, \tag{3.36}$$

where λ_i and σ_i are the singular values of $\mathbf{Y}$ and $\mathbf{X}$ respectively and s is the desired rank of the approximation [114]. In particular, considering a rank-1 approximation, it can be shown that, with high probability [114, 115]

$$(1-\epsilon)\sigma_1^2 \leq \lambda_1^2 \leq \sigma_1^2. \tag{3.37}$$

The existence of such upper and lower bounds motivates the use of K-lines clustering on $\mathbf{Y}$ instead of $\mathbf{X}$.

3.5.3 Gaussian Mixture Models for Clustering

Statistical clustering algorithms assume that the data is a realization from a mixture of probability distributions. In the case of mixture of K arbitrary distributions, the

overall probability density is given by,

$$f = \sum_{i=1}^{K} w_i f_i \quad \text{s.t.} \quad w_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{K} w_i = 1, \tag{3.38}$$

where f_i is the probability distribution function (pdf) and w_i is the non-negative weight of the i^{th} distribution.

The best SVD subspace for a spherical Gaussian distribution is any subspace through its mean [116]. More importantly, the best K-dimensional SVD subspace for a mixture of K Gaussians whose covariance is a scalar multiple of identity, contains the span of the means of the component distributions. This can also be extended to a mixture of arbitrary distributions. In general, we do not have the exact statistics of a GM and we have only the samples of realizations. The covariance matrix is also not a scalar multiple of identity. Even under these conditions, it has been proved that the SVD subspace of the sample matrix is not far from the subspace spanned by the actual component means [116].

Two spherical Gaussians $\mathcal{N}(\mu_1, \sigma_1^2\mathbf{I})$ and $\mathcal{N}(\mu_2, \sigma_2^2\mathbf{I})$ are considered to be c-separated if $\|\mu_1 - \mu_2\|_2^2 \geq c^2 M \max(\sigma_1^2, \sigma_2^2)$, where M is the dimension of the Gaussian [115]. A 2-separated mixture corresponds to almost completely separated Gaussians, whereas a 1- or 1/2-separated mixture contains Gaussians which overlap significantly. By projecting the Gaussian mixtures on to a K-dimensional subspace spanned by the means of K Gaussians, we are equivalently projecting the Gaussian mixtures onto their best rank-K SVD subspace. This preserves the distance between the means (intercluster distance), whereas the intracluster distance reduces drastically [116]. Therefore, the separation between the Gaussians in the mixture increases and the clustering performance improves. Similar results can also be shown for mixtures of Gaussians with arbitrary covariances [116].

3.5.4 Application: Video Clustering

Clustering of video frames is more than just a generalization of clustering of images. This is because the video frames that convey meaning as a group are both statistically and semantically related. One of the popular approaches to video clustering involves extracting the keyframes by shot boundary detection and clustering the keyframes together to derive a semantic interpretation [117]. However, it is important to understand that extraction of the keyframes by detecting the shot boundaries itself is a fundamental clustering problem which we address in this chapter. Video frames of a single shot have similar background structure and they can be clustered together using color histograms or distance measures.

Using the fact that the video frames that have similar backgrounds are close together in terms of the Euclidean distance measure (l_2 norm), we perform distance based clustering. It is important to clarify the notion of background in this problem. Background is the region in a frame that remains relatively motionless. Even if some objects in the foreground are relatively motionless they can be treated as background. We use both the K-means and the K-lines clustering algorithms for clustering the video data. We consider four different approaches: a) basic K-means/K-lines clustering on the high dimensional data, b) reducing the dimensions of the data using random projections prior to clustering, c) reducing the dimensions of the data assuming it as a mixture of Gaussians prior to clustering and d) reducing the dimensions, first using RP and then under the mixture of Gaussians assumption, prior to clustering. Both centroid and left singular vector of a group of video frames retrieve the background information effectively. This motivates the use of both K-means/K-lines clustering in the proposed approaches. We will assume that we have K clusters of the T video frames and we vectorize each video frame into a M dimensional vector thereby generating the $M \times T$ matrix $\mathbf{X}$. The K index sets of the clusters are given by

$\{\Lambda_i\}_{i=1}^{K}$ and $T_i = |\Lambda_i|$. In the remaining part of this section, we describe the different approaches for clustering.

The RP method can be used to reduce the dimensionality of the data matrix $\mathbf{X}$ according to (3.34), preserving the length of the data vectors, pairwise distances and angles with high probability. We have also seen in Section 3.5.1 that the centroid and SVD of a set of data vectors are approximately preserved with a high probability, given a sufficiently large number of measurements N.

Assuming that $K = 1$, the centroid and the first singular vector of $\mathbf{Y}$ are approximately equal to that of $\mathbf{X}$ for a sufficiently large N. If $K > 1$, the centroid and first singular vector of each cluster will still be approximately preserved because $T_i < T$. Therefore, the RP method will be useful regardless of the number of clusters, provided we choose N based on the assumption of single cluster. The linear increase in the number of data vectors T will not change N significantly because $N \propto \ln T$. Therefore, in order to perform RP based clustering, we use either the K-means or the K-lines clustering algorithm on the reduced dimensional data.

We know from Section 3.5.3 that the rank-K SVD subspace of the sample matrix is not far from the space spanned by the K component means even when the Gaussians are not spherical. In this approach for clustering, we assume that $\mathbf{X}$ contains realizations from a mixture of K Gaussians, not necessarily spherical, where each Gaussian represents a cluster. We compute the best rank-K subspace of $\mathbf{X}$ using SVD and denote the basis vectors of the rank-K subspace (first K left singular vectors of $\mathbf{X}$) by $\mathbf{U}_K$. The projection to the rank-K subspace is given by,

$$\mathbf{W} = \mathbf{U}_K^T \mathbf{X}, \tag{3.39}$$

where $\mathbf{W}$ contains the K dimensional data vectors after projection. Because of the reasoning provided in Section 3.5.3, the clusters in the K dimensional space are more separated than the clusters in the M dimensional space. Therefore, we perform K-

Table 3.1: Algorithm to cluster video data and identify keyframes.

Goal: To perform clustering of high-dimensional long video sequences using the approach given in Section 3.5.4

Variables

High-dimensional data matrix, $\mathbf{X}$ of size $M \times T$.
Intermediate data matrix after RP, $\mathbf{Y}$ of size $N \times T$.
Final data matrix used for clustering, $\mathbf{Z}$ of size $K \times T$.
Initial number of clusters, J.
Actual number of clusters, K.
Cluster centroid matrix (J Clusters), $\mathbf{B}$ of size $K \times J$.
Cluster centroid matrix (K Clusters), $\mathbf{A}$ of size $K \times K$.
Index set for the i^{th} cluster, $\mathbf{\Lambda}_i$.
Gaussian i.i.d. random matrix, $\mathbf{R}$ of size $M \times N$.

Algorithm

1. Compute the RP, $\mathbf{Y} = (1/\sqrt{N})\mathbf{R}^T\mathbf{X}$.
2. Compute rank-K SVD, $[\mathbf{U}_K, \mathbf{S}_K, \mathbf{V}_K] = \text{SVD}(\mathbf{Y}, K)$.
3. Project on to the K-dimensional SVD space, $\mathbf{Z} = \mathbf{U}_K^T\mathbf{Y}$.
4. Initialize $\mathbf{B}$ with J randomly chosen columns of $\mathbf{Z}$.
5. Perform K-means/K-lines clustering for J clusters.
6. Using greedy combinations of columns of $\mathbf{B}$, create $\mathbf{A}$.
7. Perform K-means/K-lines clustering for K clusters using $\mathbf{A}$ as initial centroids.
8. Using the index sets $\mathbf{\Lambda}_i$ obtained from clustering, identify the keyframes.

means or K-lines clustering on $\mathbf{W}$ and identify the index sets of the clusters. Similar to the previous case, in this approach we assume $\mathbf{X}$ to be a set of T realizations of K Gaussians. Furthermore, as RP approximately preserves the pairwise distances of the samples, realizations from a mixture of K Gaussians in M dimensions preserve their structure in N dimensions. This fact is used to further reduce the computational complexity of the framework.

In this approach, we first project $\mathbf{X}$ to obtain $\mathbf{Y}$ according to (3.34). The elements of $\mathbf{Y}$ are treated as realizations of a mixture of K Gaussians in N dimensions. From the arguments in Section 3.5.3, we project $\mathbf{Y}$ onto a K dimensional SVD subspace to obtain $\mathbf{Z}$. K-means or K-lines clustering can be performed on $\mathbf{Z}$ to iden-

Figure 3.4: Keyframes obtained by clustering the test data using the algorithm in Table 3.1.

tify the index sets of the clusters. Note that in this case, we first perform an initial level of dimensionality reduction using RP, which aids in a faster computation of the SVD subspace. In the second stage, we reduce the dimensionality further using the GM assumption. Hence, this approach combines the advantages of both the previous approaches in terms of a much reduced computational complexity due to RP and improved clustering performance along with further reduction in computational complexity due to the assumption of Gaussian mixtures. The outline of the algorithm to perform video clustering and identify the keyframes is shown in Table 3.1.

To improve the clustering performance, we adopt a two stage approach to clustering. This reduces the possibility of the clustering algorithm being stuck in a local minima. Initially, we solve the clustering problem for a number of clusters J that is larger than the actual number K. Then, we greedily combine the columns of the cluster centroid matrix $\mathbf{B}$ in order to obtain the matrix $\mathbf{A}$ (step 6 of Table 3.1). In this greedy combination method we first choose the two most similar vectors of $\mathbf{B}$ and combine them. We repeat this procedure for two columns at a time until we are left with only K columns. In the K-means method two most similar vectors are the ones that have the minimum pairwise Euclidean distance and the combined vector is the mean of the two. In K-lines clustering two most similar vectors are the ones

that have the maximum correlation and the combined vector is principal left singular vector of the matrix of the two vectors.

3.5.5 Simulation Results

The video sequences in QCIF format, used for evaluating the performance of the algorithms, were obtained from [118] and the spatial resolution was changed to 128×128. The first test data set was generated by stitching 10 different video sequences and it contains 1900 frames in total. The second test data set has a total of 550 frames obtained from 3 different video sequences.The initial number of clusters J is set to 3 times the actual number of clusters K. For the first data set $K = 10$ and for the second data set $K = 3$. The keyframes are identified using all the four approaches for both K-means and K-lines clustering. The keyframes obtained for the first data set are shown in Figure 3.4. The keyframes identified are similar with the all the four approaches and we also obtain 100% clustering performance.

The running times for the different approaches in MATLAB (version R2007b) to cluster the test data are listed in Table 3.2. It can be seen that the approach based on RP and GM is of least computational complexity. The running time for the K-means and the K-lines clustering algorithm are close to each other except for the case of the basic approach, which however is not the choice when clustering high dimensional data.

3.6 Kernel K-lines Clustering

3.6.1 Linear Iterative Procedure for K-lines Clustering

Since direct computation of SVDs in feature space is computationally expensive and sometimes not even possible, we will provide a solution for cluster update using an iterative procedure which does not require computation of SVDs. For a cluster k, we will compute the set of non-zero coefficients $\{a_{nk}\}$, where $n \in \mathcal{C}_k$, and the cluster center $\boldsymbol{\psi}_k$ in an alternating manner. In the first step, we fix $\boldsymbol{\psi}_k$ and compute $\{a_{nk}\}$

Table 3.2: Comparison of running time for the different clustering approaches in MATLAB. The results for the first and the second data sets are separated by a slash (/). The third approach could not run for the first data set owing to memory issues because of high dimensionality.

Approach	Running time(s)		Number of Dimensions
	K-means	K-lines	
Basic	696.51/37.69	774.23/102.22	16384/16384
RP	33.87/7.59	32.17/10.15	400/400
GM	-/10.18	-/10.30	10/3
RP and GM	29.41/7.36	26.84/7.45	10/3

as its least squares estimate, $a_{nk} = \mathbf{y}_n^T \boldsymbol{\psi}_k, \forall n \in \mathcal{C}_k$. Incorporating the constraint $\|\boldsymbol{\psi}_k\|_2 = 1$ and assuming $\{a_{nk}\}$ to be known, in the second step, we compute

$$\boldsymbol{\psi}_k = \sum_{n \in \mathcal{C}_k} a_{nk} \mathbf{y}_n \Big/ \left\| \sum_{n \in \mathcal{C}_k} a_{nk} \mathbf{y}_n \right\|_2 . \tag{3.40}$$

This is equivalent to computing the normalized weighted mean of data samples in a cluster. Repeating these two steps for a sufficient number of iterations will result in a good estimate for $\boldsymbol{\psi}_k$ and a_{nk}.

Membership Set Update

The optimization problem to compute the coefficients, membership sets, and cluster centers is expressed as

$$\{\boldsymbol{\Psi}, \{\mathcal{C}_k\}, \{a_{nk}\}\} = \underset{\boldsymbol{\Psi}, \{\mathcal{C}_k\}, \{a_{nk}\}}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{n \in \mathcal{C}_k} \|\mathbf{y}_n - a_{nk} \boldsymbol{\psi}_k\|_2^2$$
$$\text{subject to } \|\boldsymbol{\psi}_k\|_2^2 = 1, \ \forall k, \tag{3.41}$$

where $\{\mathcal{C}_k\}$ denotes the collection of membership sets, and $\{a_{nk}\}$ indicates the collection of non-zero coefficients. In order to estimate the cluster membership sets, we note that $\|\boldsymbol{\psi}_k\|_2^2 = 1$ and expand

$$\|\mathbf{y}_n - a_{nk} \boldsymbol{\psi}_k\|_2^2 = \mathbf{y}_n^T \mathbf{y}_n + a_{nk}^2 - a_{nk} \mathbf{y}_n^T \boldsymbol{\psi}_k. \tag{3.42}$$

Substituting this in (3.41) and neglecting $\mathbf{y}_n^T\mathbf{y}_n$, since it does not affect the computation of the collection of membership sets $\{\mathcal{C}_k\}$, where $k = \{1, \ldots, K\}$, we obtain

$$\{\mathcal{C}_k\} = \underset{\{\mathcal{C}_k\}}{\operatorname{argmax}} \sum_{k=1}^{K} \sum_{n \in \mathcal{C}_k} (\mathbf{y}_n^T \boldsymbol{\psi}_k)^2, \tag{3.43}$$

where the cluster centers $\boldsymbol{\Psi}$ are fixed and using the least squares estimates for the coefficients $\{a_{nk}\}$, where $n \in \mathcal{C}_k$, is used. Eqn. (3.43) can be recast according to Lemma 3.6.1.

Lemma 3.6.1 *In K-hyperline clustering, when the cluster centers are fixed, computing the collection of membership sets using (3.43) can be rewritten as*

$$\{\mathcal{C}_k\} = \underset{\{\mathcal{C}_k\}}{\operatorname{argmax}} \sum_{k=1}^{K} \sum_{n \in \mathcal{C}_k} |\mathbf{y}_n^T \boldsymbol{\psi}_k|. \tag{3.44}$$

Proof Eqn. (3.43) can be expressed as,

$$\{\mathcal{C}_k\} = \underset{\{\mathcal{C}_k\}}{\operatorname{argmax}} \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}(n \in \mathcal{C}_k)(\mathbf{y}_n^T \boldsymbol{\psi}_k)^2, \tag{3.45}$$

where $\mathbb{I}$ is the indicator function that returns the value 1 when its argument is true and zero otherwise. From (3.45), the cluster membership for a single data sample $\mathbf{y}_n$ is given by the index

$$k = \underset{t}{\operatorname{argmax}} (\mathbf{y}_n^T \boldsymbol{\psi}_t)^2. \tag{3.46}$$

The function $\mathbf{y}_n^T\boldsymbol{\psi}_t$ evaluated for $t = \{1, \ldots, K\}$ results in K discrete values for a given $\mathbf{y}_n$. Note that the index k that results in the maximum squared value of the function, $(\mathbf{y}_n^T\boldsymbol{\psi}_k)^2$, will be the same as the one that results in the maximum absolute value, $|\mathbf{y}_n^T\boldsymbol{\psi}_k|$. Hence, replacing $(\mathbf{y}_n^T\boldsymbol{\psi}_k)^2$ with $|\mathbf{y}_n^T\boldsymbol{\psi}_k|$ in (3.45) and subsequently in (3.43), will not change the membership sets computed. Therefore, (3.44) holds true.

Based on Lemma (3.6.1), the collection of membership sets $\{\mathcal{C}_k\}$, where $k = \{1, \ldots, K\}$ can be updated using (3.44).

3.6.2 Clustering Procedure using Matrix Operations

Expressing the K-hyperline clustering in terms of matrix operations, will allow us to conveniently formulate the feature space version of the problem. Let us first define the membership matrix of the data samples by $\mathbf{Z} \in \mathbb{R}^{N \times K}$, where $z_{nk} = 1$ if and only if $n \in \mathcal{C}_k$. The coefficient matrix is denoted as $\mathbf{A} \in \mathbb{R}^{N \times K}$, where each element $a_{nk} = \mathbf{y}_n^T \boldsymbol{\psi}_k$. Therefore, cluster assignment is performed by computing $\mathbf{A} = \mathbf{Y}^T \boldsymbol{\Psi}$, and then setting $\mathbf{Z} = g(\mathbf{A})$, where $g(.)$ returns 1 at the location of absolute maximum of each row of the argument matrix and zero elsewhere. Note that this membership update step is the same as that described in Section 3.6.1, except that it is defined using matrices here. Denoting $\mathbf{H} = \mathbf{Z} \odot \mathbf{A}$, the centroid is updated as $\boldsymbol{\Psi} = \mathbf{YHD}$, where $\odot$ indicates the Hadamard product and $\mathbf{D}$ is a diagonal matrix that normalizes the columns of $\boldsymbol{\Psi}$.

3.6.3 Algorithm

The kernel K-hyperline clustering performs clustering in the feature space using Mercer kernels. Let us define the non-linear transformation to a feature space $\mathcal{F}$ as $\Phi : \mathbb{R}^M \mapsto \mathcal{F}$. Since the Hilbert space $\mathcal{F}$ may be very high-dimensional (even infinite), all the operations in the space will be carried out exclusively using inner products defined by the kernel function $K(\mathbf{y}_i, \mathbf{y}_j) = \Phi(\mathbf{y}_i)^T \Phi(\mathbf{y}_j)$. Note that the possibility of providing closed form expressions for Φ depends on the form of the kernel used. For example, in the case of a third degree polynomial kernel expressed as $K(\mathbf{y}_i, \mathbf{y}_j) = (\mathbf{y}_i^T \mathbf{y}_j)^3$, if we assume $\mathbf{y}_i = [y_{i,1}\ y_{i,2}]^T$, then we have $\Phi(\mathbf{y}_i) = [y_{i,1}^3\ \sqrt{3} y_{i,1}^2 y_{i,2}\ \sqrt{3} y_{i,1} y_{i,2}^2\ y_{i,2}^3]^T$. It is clear that the dimension of $\mathcal{F}$ will increase rapidly as the dimension of the data vector $\mathbf{y}_i$ increases. The transformation $\Phi(.)$ results in an infinite dimensional Hilbert space in the case of radial basis function (RBF) kernel defined as

$$K(\mathbf{y}_i, \mathbf{y}_j) = \exp(-0.5 \|\mathbf{y}_i - \mathbf{y}_j\|_2^2). \tag{3.47}$$

Table 3.3: The Kernel K-lines Clustering Algorithm.

Input
$\mathbf{Y} = [\mathbf{y}_i]_{i=1}^N$, $M \times N$ matrix of data samples.
$\mathbf{K_{YY}}$, $N \times N$ kernel matrix.
K, desired number of clusters.

Initialization
- Randomly group data samples into K clusters and initialize the membership matrix $\mathbf{Z}$.
- Based on $\mathbf{Z}$, obtain the rank-1 SVD for each cluster to initialize $\mathbf{\Psi}$.
- Compute the initial coefficients, $\mathbf{A} = \mathbf{Y}^T\mathbf{\Psi}$.

Algorithm
Loop until convergence
  Loop for L iterations
    - Compute $\mathbf{H} = \mathbf{Z} \odot \mathbf{A}$.
    - Compute $\mathbf{A} = \mathbf{K_{YY}H\Gamma}(\Phi(\mathbf{Y})\mathbf{H})^{-1}$.
  end
  - Update the membership matrix $\mathbf{Z}$ by identifying the index of absolute maximum in each row of $\mathbf{A}$.
end

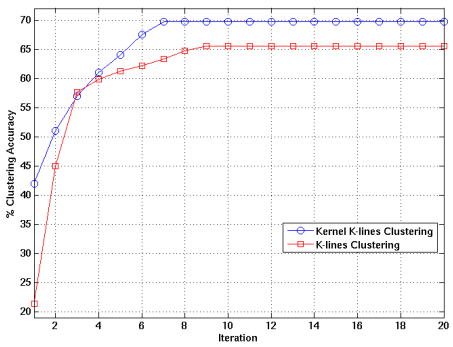
Let $\Phi(\mathbf{Y})$ and $\Phi(\mathbf{\Psi})$ denote the N transformed feature vectors and K transformed cluster centroid matrices respectively. The kernel similarity matrix of inner products is constructed as $\mathbf{K} = \Phi(\mathbf{Y})^T\Phi(\mathbf{Y})$ and $K(\mathbf{y}_i, \mathbf{y}_j)$ is its $(i,j)^{\text{th}}$ element. The coefficient matrix $\boldsymbol{\alpha}$ in the feature space can be computed as $\boldsymbol{\alpha} = \Phi(\mathbf{Y})^T\Phi(\mathbf{\Psi})$ and the membership matrix in the feature space is $\mathbf{Z} = g(\boldsymbol{\alpha})$. Hence, the cluster centers in the feature space are computed as

$$\Phi(\mathbf{\Psi}) = \Phi(\mathbf{Y})\mathbf{HN_\Psi}, \tag{3.48}$$

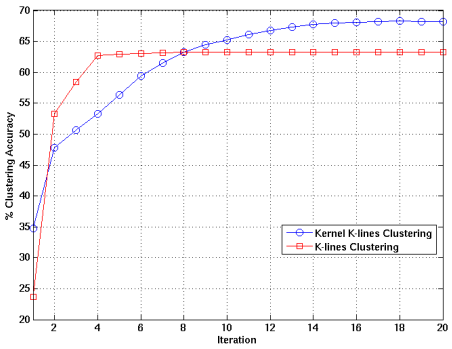
where $\mathbf{H} = \mathbf{Z} \odot \boldsymbol{\alpha}$. The normalization term can be obtained as

$$\mathbf{N_\Psi} = [diag((\Phi(\mathbf{Y})\mathbf{H})^T\Phi(\mathbf{Y})\mathbf{H})]^{-1/2} = [diag(\mathbf{H}^T\mathbf{K}\mathbf{H})]^{-1/2}, \tag{3.49}$$

where $diag(.)$ returns a diagonal matrix with the diagonal elements same as that of



(a) Pendigits dataset.



(b) USPS dataset.

Figure 3.5: Performance of Kernel K-lines clustering algorithm with two different datasets over multiple iterations. In each case, the performance of K-lines clustering algorithm is also shown.

the argument. Combining $\boldsymbol{\alpha} = \Phi(\mathbf{Y})^T\Phi(\boldsymbol{\Psi})$ and (3.48), we obtain

$$\boldsymbol{\alpha} = \Phi(\mathbf{Y})^T\Phi(\mathbf{Y})\mathbf{H}\mathbf{N}_{\boldsymbol{\Psi}} = \mathbf{K}\mathbf{H}\mathbf{N}_{\boldsymbol{\Psi}}. \tag{3.50}$$

Hence, kernel K-hyperline clustering can be performed by updating $\boldsymbol{\alpha}$ using (3.50) and $\mathbf{H} = \mathbf{Z} \odot \boldsymbol{\alpha}$ in every iteration whereas the membership matrix $\mathbf{Z} = g(\boldsymbol{\alpha})$ is updated once every L iterations. The steps of the algorithm are presented in Table 3.3.

In order to demonstrate the behavior of the proposed kernel K-lines clustering procedure, we performed clustering of the data samples from two benchmark datasets: *USPS* and *Pendigits* obtained from the UCI database [119]. No preprocessing was performed on the datasets and we chose the RBF kernel with $\gamma = \frac{1}{32}$ to evaluate the similarities between the samples. It can be clearly seen from Figure 3.5 that the kernel K-lines clustering approach achieves higher clustering accuracies in comparison to using K-lines clustering on both the datasets.

Chapter 4

DISCRIMINATIVE CLUSTERING FOR MIXING MATRIX ESTIMATION

4.1 Problem Statement

Blind source separation (BSS) is the problem of estimating the original source signals from their mixtures, when the mixing process is unknown. In instantaneous blind source separation, we typically observe P-channel sensor signals given by the rows of the observation matrix $\mathbf{C} \in \mathbb{R}^{P \times N}$. Here, $\mathbf{C}$ is obtained as the linear combination of R unknown sources as

$$\mathbf{C} = \mathbf{G}\mathbf{S} + \mathbf{N}, \tag{4.1}$$

where $\mathbf{S} \in \mathbb{R}^{R \times N}$ is the set of R source signals given by the rows of the matrix, $\mathbf{G} \in \mathbb{R}^{P \times R}$ is the mixing matrix and $\mathbf{N} \in \mathbb{R}^{P \times N}$ is the matrix denoting the corruption by additive Gaussian noise.

Several algorithms have been proposed to estimate the mixing matrix and the source signals using only the knowledge of the noisy observations [120]. When the source signals are sparse either in the measured domain or some parametric domain, the columns of the observed data matrix $\mathbf{C}$ will be concentrated along the directions of the columns of the unknown mixing matrix $\mathbf{G}$ [121]. Hence, the columns of $\mathbf{C}$ can be clustered using standard clustering algorithms and the set of cluster centers will form the estimate of $\mathbf{G}$. Given the mixing matrix, the set of unknown sources can be estimated using two different approaches. When the number of sensors is greater than the total number of sources ($P > R$), the problem of estimating the sources is referred to as overdetermined BSS and Independent Component Analysis (ICA) can be used to estimate the sources. However in underdetermined BSS, i.e., when there are more sources than the number of sensors ($R > P$), the performance of ICA based methods is often affected adversely. In this case, sparse component analysis (SCA) methods that exploit the sparse representability of the sources, instead of their independence,

can be used for separating the mixture components. Sparsity of a vector means that only a small number of its elements differ significantly from zero. For SCA methods to work, the observation vectors should be a sparse linear combination of the columns of the mixing matrix. This implies that the columns of the source matrix must be sparse, with respect to some linear transformation. Note that, the source/observed signals can be sparsely decomposed either into a pre-defined signal dictionary, such as the Fourier transform, or overcomplete dictionaries can be adapted to the data [83]. Hence it is clear that sparsity of sources in some known domain is necessary for both mixing matrix estimation and subsequently source separation using SCA methods.

In this work, we consider the problem of estimating $\mathbf{G}$ in both overdetermined and underdetermined mixing conditions. Though the methods proposed in this work are general in application, we demonstrate their use in mixing matrix estimation from speech mixtures. Typically separation of speech sources is performed in the time-frequency (TF) domain or a parameteric domain, since speech is not sparse in the time domain [122], [123], [124]. The DUET-type (Degenerate Unmixing Estimation Technique) methods reported in [125], [126] can be used to estimate the mixing matrix effectively when the W-disjoint orthogonality condition is approximately satisfied across the whole TF region. The TIFROM-type (Time-frequency Ratio of Mixtures) methods proposed in [127], [128] allow overlapping of sources to a certain degree in the TF regions, but still require adjacent regions where only one source is active. In practical situations, the conditions required by these methods are not satisfied. Hence, the authors in [129] relaxed these requirements and proposed an estimation method based on TF ratios. Mixing matrix estimation can also be performed using standard clustering algorithms and the K-lines clustering algorithm proposed in [23] performs significantly better than other procedures. However, the complexity of clustering increases significantly when the number of data samples increases.

The mixing matrix estimation is performed by applying clustering algorithms on a subset of observations in the TF domain where only one source is active. These points are referred to as single source points (SSPs) [130]. The source matrix column corresponding to an SSP will have only one non-zero element, i.e., it will be $1-$sparse. Using SSPs allows us to obtain high mixing matrix estimation performance just using clustering based approaches. Restating the model in (4.1) such that it represents the sources and mixtures in the TF domain, we have $\mathbf{c}(t,k) = \mathbf{G}\mathbf{s}(t,k)$, under zero noise conditions. Here, $\mathbf{s}(t,k)$ and $\mathbf{c}(t,k)$ respectively denote the complex source and observation vectors at the time index t and the frequency index k. The direction of the modulus of the observation vectors, $\mathbf{c}(t,k)$, in the SSPs where only one of the sources is active, will be same as those of the columns of $\mathbf{G}$ [130]. However, the probability of identifying such TF points is extremely low, particularly when there are multiple sources and non-negligible noise. Hence, the authors in [3] developed an algorithm that identifies approximate SSPs and employs a hierarchical clustering procedure to estimate the columns of $\mathbf{G}$. This method achieved high mixing matrix estimation performance, with multiple speech sources, under different mixing conditions. However, it is possible to replace the hierarchical clustering procedure [3] with more suitable clustering algorithms to improve the performance.

In this work, we develop two algorithms to perform discriminative clustering of the SSPs in underdetermined and overdetermined mixing conditions respectively. Both the proposed methods iteratively project the SSPs onto a subspace optimized for discrimination and perform K-lines clustering in that subspace. In overdetermined mixing conditions, we have $P > R$, and a low-dimensional linear subspace of $R-1$ dimensions can be identified for performing discriminative clustering. However, in underdetermined mixing, we have $R > P$ and hence it is not possible to identify such a subspace. Therefore, we propose to project the SSPs onto a high-dimensional feature space prior to performing discriminative clustering in this case. Using the proposed

algorithms, we demonstrate improvements in mixing matrix estimation performance when compared to the methods in [129], [23] and [3].

4.2 Need for Discriminative Clustering

The performance of a clustering algorithm depends on (a) the suitability of the clustering cost to the data and, (b) the separability of the classes with respect to the distortion function. The clustering performance can be improved by increasing the linear separability between the classes prior to clustering data. In supervised classification it is typical to learn a linear discriminant function, that maps the data samples to a subspace, in order to improve the linear separability between classes. For some types of data, it may not be possible to improve the class separability using a linear discriminant function. Non-linearly mapping the data to a feature space and identifying a discriminant subspace of the feature space is necessary in such cases. Usually, the feature space is of very high dimensions but since it is equipped with a reproducing kernel, it is possible to use the *kernel trick* to perform computations efficiently in that space [131].

Since clustering is unsupervised, determining a discriminant function is not straightforward. The discriminative clustering frameworks for K-means proposed in [132], [133], [134], [135] iteratively perform clustering to evaluate the class labels for the data samples and linear discriminant analyis (LDA) to identify the discriminant function for maximal linear separability. Though discriminative K-means clustering has achieved high clustering accuracies with standard datasets, it is not well suited for mixing matrix estimation in blind source separation. In this work, we propose the K-lines-LDA (Khyp-LDA) algorithm to perform discriminative K-lines clustering by iteratively computing the linear mapping for maximal discrimination. We employ a modified LDA procedure that considers the between-class and within-class similarities, instead of the scatter measures traditionally used in LDA. The Khyp-LDA

algorithm can be employed only in overdetermined mixing conditions as it projects the $P-$dimensional data onto $R-1$ dimensions ($P > R$) and performs clustering to estimate the columns of the mixing matrix.

In order to extend this to underdetermined BSS, we propose to non-linearly map the data onto a feature space prior to performing discriminative K-lines clustering. The generalized discriminant analysis (GDA) proposed in [136] can be used to identify the linear subspace of the feature space obtained by a non-linear mapping, for maximal discrimination. We develop the K-lines-GDA (Khyp-GDA) algorithm that iterates between the modified GDA with similarity measures, and kernel K-lines clustering, that works with the feature space data projected onto the linear subspace identified by GDA.

Simulations with synthetic data show that the proposed approaches achieve improved mixing matrix estimation performance at different levels of source sparsity and disjoint orthogonality in the source matrix. Furthermore, we employed the proposed discriminative K-lines clustering algorithms in order to estimate the mixing matrix columns from the SSPs obtained using the algorithm in [3]. Results for estimation of $\mathbf{G}$ from mixtures of multiple speech sources show that the proposed algorithms achieve much improved estimation performance in comparison to the approaches in [129], [23], and [3]. The improvements were observed for both underdetermined and overdetermined mixing, under different noise conditions.

4.3 Background

In this section, we describe the estimation of SSPs from the TF representation of the mixtures [3] and briefly review linear discriminant based clustering algorithms. Throughout this chapter, we will denote matrices using bold upper case, vectors using bold lower case and scalars using upper or lower case.

4.3.1 Identification of SSPs

We are interested in performing mixing matrix estimation from the time-frequency representation of the mixtures. As described earlier, SSPs are TF points at which only one of the sources is active. As an example scenario, let us consider the case with 2 sources and 2 sensors. Let $\mathbf{g}_1$ and $\mathbf{g}_2$ denote the columns of the mixing matrix $\mathbf{G} \in \mathbb{R}^{2\times 2}$. At an SSP (t_1, k_1) in the TF domain where only the first source is active, i.e., $s_1(t_1, k_1) \neq 0$ and $s_2(t_1, k_1) = 0$, we have

$$\mathbf{c}(t_1, k_1) = \mathbf{g}_1 s_1(t_1, k_1). \tag{4.2}$$

This implies that the real and imaginary parts can be equated as

$$Re\{\mathbf{c}(t_1, k_1)\} = \mathbf{g}_1 Re\{s_1(t_1, k_1)\}, \tag{4.3}$$

$$Im\{\mathbf{c}(t_1, k_1)\} = \mathbf{g}_1 Im\{s_1(t_1, k_1)\}. \tag{4.4}$$

Here, it can be easily seen that the absolute directions of the real and imaginary components of $\mathbf{c}(t_1, k_1)$ are the same as that of $\mathbf{g}_1$. Now, consider another time-frequency point (t_2, k_2) where both the sources are active.

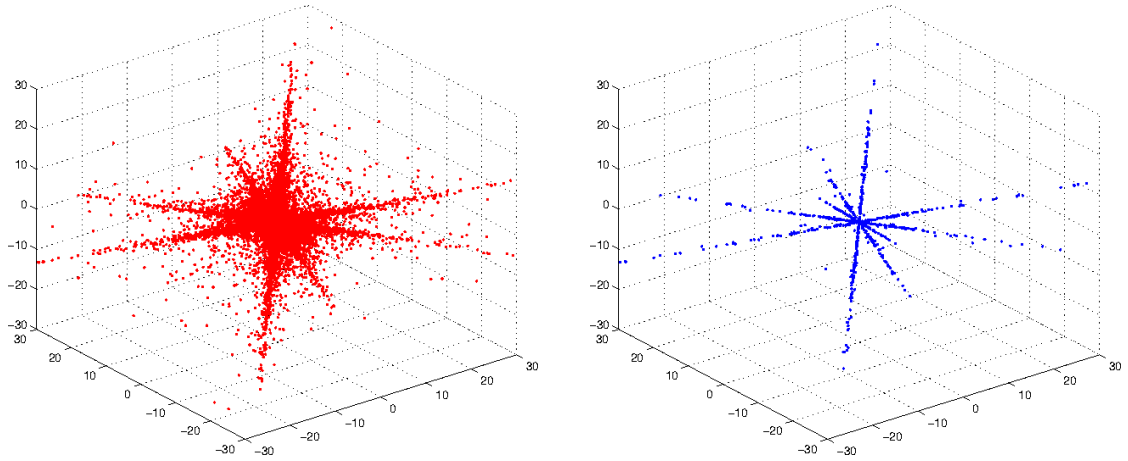
$$Re\{\mathbf{c}(t_2, k_2)\} = \mathbf{g}_1 Re\{s_1(t_2, k_2)\} + \mathbf{g}_2 Re\{s_2(t_2, k_2)\}, \tag{4.5}$$

$$Im\{\mathbf{c}(t_2, k_2)\} = \mathbf{g}_1 Im\{s_1(t_2, k_2)\} + \mathbf{g}_2 Im\{s_2(t_2, k_2)\}. \tag{4.6}$$

For the absolute directions of the real and imaginary components to be same in this case, the following relation needs to be satisfied,

$$\frac{Re\{s_1(t_2, k_2)\}}{Im\{s_1(t_2, k_2)\}} = \frac{Re\{s_2(t_2, k_2)\}}{Im\{s_2(t_2, k_2)\}}. \tag{4.7}$$

Extending this to the case of multiple sources, for the absolute directions of $Re\{\mathbf{c}(t, k)\}$ and $Im\{\mathbf{c}(t, k)\}$ at any TF point (t, k) to be the same, it should be a single source point or the ratios of the real and imaginary components of the short-term Fourier transform (STFT) of all source signals must be the same. However, the



(a) (b)

Figure 4.1: Distribution of the observed samples for mixing matrix estimation in underdetermined BSS, with $P = 3$ and $R = 5$: (a) All STFT coefficients, (b) SSPs identified, with $\Delta\theta = 0.8$, using the algorithm proposed in [3].

probability of finding such time frequency points will be extremely low, particularly when there are multiple sources. Hence, the authors in [3] developed an algorithm that relaxes the condition for identifying SSPs. The points in the TF domain where the angle between the absolute directions of $Re\{\mathbf{c}(t,k)\}$ and $Im\{\mathbf{c}(t,k)\}$ is less than a pre-defined threshold $\Delta\theta$ is considered as an SSP. This condition is expressed as

$$\left| \frac{Re\{\mathbf{c}(t,k)\}^T Im\{\mathbf{c}(t,k)\}}{\|Re\{\mathbf{c}(t,k)\}\|_2 \|Im\{\mathbf{c}(t,k)\}\|_2} \right| > \cos(\Delta\theta). \tag{4.8}$$

Typically, both the real and the imaginary parts of the SSPs, identified using the condition (4.8), are stacked together in order to estimate the mixing matrix. Figure 4.1 illustrates an example case of 5 sources and 3 sensors, where the entire set of time-frequency coefficients (both real and imaginary components) for the observations is shown in Figure 4.1(a) and those of the SSPs identified using (4.8) are shown in Figure 4.1(b). As it can be obviously seen, clustering the SSPs to identify the columns of $\mathbf{G}$ will be comparatively easier. Furthermore, the complexity of clustering is reduced since the number of data samples to be used for clustering is significantly reduced.

4.3.2 Linear Discriminant Based Clustering

Clustering algorithms can be combined with linear discriminant analysis (LDA) in order to improve the separability of the data belonging to different classes, thereby improving the clustering performance. The LDA algorithm computes a set of discriminant directions $\mathbf{W} \in \mathbb{R}^{M \times d}$, so that the class separability of the projected data $\mathbf{X} = \mathbf{W}^T\mathbf{Y}$ is improved. In order to find $\mathbf{W}$, the within-class and between-class scatter matrices are computed. The within-class scatter matrix,

$$\mathbf{S}_w = \sum_{k=1}^{K} \sum_{n \in \mathcal{C}_k} (\mathbf{y}_n - \boldsymbol{\mu}_k) (\mathbf{y}_n - \boldsymbol{\mu}_k)^T , \tag{4.9}$$

computes the scatter of data samples around their respective class mean vectors denoted by $\boldsymbol{\mu}_k$ for classes $k = \{1, \ldots, K\}$. Assuming that all data samples are centered to zero mean, the between-class scatter is

$$\mathbf{S}_b = \sum_{k=1}^{K} N_k \boldsymbol{\mu}_k \boldsymbol{\mu}_k^T, \tag{4.10}$$

where N_k denotes the total number of samples in the class K. Here, $d = K - 1$ and the objective is expressed as

$$\max_{\mathbf{W}} Tr \frac{\mathbf{W}^T \mathbf{S}_b \mathbf{W}}{\mathbf{W}^T \mathbf{S}_w \mathbf{W}}. \tag{4.11}$$

Note that the K-means clustering algorithm minimizes the distortion metric $Tr(\mathbf{S}_w)$ or equivalently maximizes $Tr(\mathbf{S}_b)$, where $Tr(.)$ is the trace of the argument matrix. The algorithm proposed in [132] combines K-means and LDA, by iterating between generation of class labels using K-means on the projected data $\mathbf{X}$, and computation of discriminant directions $\mathbf{W}$ based on the class labels and the $M-$dimensional data $\mathbf{Y}$.

4.4 Mixing Matrix Estimation in Overdetermined BSS

In this section, we develop the modified linear discriminant analysis using similarity measures and describe the proposed Khyp-LDA algorithm for estimating the mixing matrix in overdetermined BSS.

4.4.1 Linear Discrimination using Similarity Measures

In order to incorporate LDA into K-lines clustering, we have to modify the classical LDA procedure described in Section 4.3.2 so that similarity measures are used instead of scatter matrices.

We will consider only the computation of membership sets as given in (3.44). In order to represent this without using absolute values, we define a new set of cluster centroids given by $\mathbf{\Omega} = \{\boldsymbol{\omega}_l\}_{l=1}^{2K}$, where

$$\boldsymbol{\omega}_{2k-1} = +\boldsymbol{\psi}_k \text{ and } \boldsymbol{\omega}_{2k} = -\boldsymbol{\psi}_k, \tag{4.12}$$

for $k = \{1, \ldots, K\}$. Any data sample $\mathbf{y}_n$ that results in the maximum correlation $\mathbf{y}_n^T\boldsymbol{\omega}_l$, is associated to the cluster center $\boldsymbol{\omega}_l$. The membership set $\mathcal{M}_l,\ l = \{1, \ldots, 2K\}$ contains the indices of training vectors associated to $\boldsymbol{\omega}_l$. Now, computation of the membership sets can be obtained by modifying (3.44) as

$$\{\mathcal{M}_l\}_{l=1}^{2K} = \underset{\{\mathcal{M}_l\}}{\text{argmax}} \sum_{l=1}^{2K} \sum_{n \in \mathcal{M}_l} \mathbf{y}_n^T\boldsymbol{\omega}_l. \tag{4.13}$$

Note that the membership sets for $\mathbf{\Psi}$ can be obtained from the membership sets of $\mathbf{\Omega}$ as $\mathcal{C}_k = \mathcal{M}_{2k-1} \cup \mathcal{M}_{2k}$, for $k = \{1, \ldots, K\}$. During actual implementation, the membership sets $\{\mathcal{C}_k\}_{k=1}^{K}$ are computed using (3.44) and the membership sets $\{\mathcal{M}_l\}_{l=1}^{2K}$ are computed as follows: the element $n \in \mathcal{C}_k$ is appended to the set $\mathcal{M}_{2k-1}$ if $\mathbf{y}_n^T\boldsymbol{\psi}_k$ is positive, else n is appended to $\mathcal{M}_{2k}$.

In order to formulate an LDA procedure using similarity measures, let us define the total symmetric within-class similarity measure as

$$\mathbf{D}_w = \sum_{l=1}^{2K} \sum_{n \in \mathcal{M}_l} \left(\mathbf{y}_n\boldsymbol{\omega}_l^T + \boldsymbol{\omega}_l\mathbf{y}_n^T \right). \tag{4.14}$$

Note that the double summation in (4.13) can be replaced using $Tr(\mathbf{D}_w)$. The overall cluster center, $\boldsymbol{\psi}$, is computed as the left singular vector corresponding to the largest singular value of $\mathbf{Y}$ and let us also define $\boldsymbol{\omega}_p = +\boldsymbol{\psi}$ and $\boldsymbol{\omega}_m = -\boldsymbol{\psi}$. Let the set $\mathcal{M}_p$

contain the indices of the cluster centers $\boldsymbol{\psi}_k$ that have positive correlations with $\boldsymbol{\omega}_p$ and the set $\mathcal{M}_m$ contain the indices of the remaining cluster centers. Analogous to the within-class similarity, we define the symmetric between-class similarity as

$$\mathbf{D}_b = \sum_{k \in \mathcal{M}_p} (\boldsymbol{\psi}_k \boldsymbol{\omega}_p^T + \boldsymbol{\omega}_p \boldsymbol{\psi}_k^T) + \sum_{k \in \mathcal{M}_m} (\boldsymbol{\psi}_k \boldsymbol{\omega}_m^T + \boldsymbol{\omega}_m \boldsymbol{\psi}_k^T). \tag{4.15}$$

In order to compute the linear discriminant, we consider the linear transformation $\mathbf{W}$ that projects the training data $\mathbf{y}$ and cluster center $\boldsymbol{\omega}$ to the lower dimensional $\mathbb{R}^d$ space, as $\mathbf{W}^T\mathbf{y}$ and $\mathbf{W}^T\boldsymbol{\omega}$ respectively. From (4.14) and (4.15), the similarity measures in the $\mathbb{R}^d$ space can be written as

$$\mathbf{D}_w^W = \mathbf{W}^T \mathbf{D}_w \mathbf{W}, \qquad \mathbf{D}_b^W = \mathbf{W}^T \mathbf{D}_b \mathbf{W}. \tag{4.16}$$

Since we are interested in maximizing the within-class similarity and minimizing the between-class similarity, the objective for computing $\mathbf{W}$ is

$$\max_{\mathbf{W}} Tr \left((\mathbf{W}^T \mathbf{D}_b \mathbf{W})^{-1} \mathbf{W}^T \mathbf{D}_w \mathbf{W} \right). \tag{4.17}$$

Using the procedure in [137], it can be shown that the LDA directions can be updated with the eigenvectors corresponding to the d largest eigenvalues of the matrix $\mathbf{D}_b^{-1}\mathbf{D}_w$. Note that the total within- and between-class similarities in (4.14) and (4.15) can also be expressed as

$$\mathbf{D}_w = \mathbf{Y}\mathbf{M}\boldsymbol{\Omega}^T + \boldsymbol{\Omega}\mathbf{M}^T\mathbf{Y}, \tag{4.18}$$

$$\mathbf{D}_b = (\boldsymbol{\Psi}\mathbf{M}_p\boldsymbol{\omega}_p^T + \boldsymbol{\omega}_p\mathbf{M}_p^T\boldsymbol{\Psi}^T) + (\boldsymbol{\Psi}\mathbf{M}_m\boldsymbol{\omega}_m^T + \boldsymbol{\omega}_m\mathbf{M}_m^T\boldsymbol{\Psi}^T). \tag{4.19}$$

The membership matrix $\mathbf{M} \in \mathbb{R}^{N \times 2K}$ is defined for the set of cluster centers $\boldsymbol{\Omega}$, such that if $n \in \mathcal{M}_l$, we have $m_{nl} = 1$, else $m_{nl} = 0$. Similarly, $\mathbf{M}_p \in \mathbb{R}^{K \times 1}$ and $\mathbf{M}_m \in \mathbb{R}^{K \times 1}$ are defined for the membership sets $\mathcal{M}_p$ and $\mathcal{M}_m$ respectively.

4.4.2 Khyp-LDA Algorithm

The Khyp-LDA algorithm, presented in Table 4.1, is an iterative discriminative clustering procedure that combines the K-lines clustering along with the discriminant

Table 4.1: Khyp-LDA clustering for mixing matrix estimation in overdetermined BSS.

Input
$\mathbf{Y} = [\mathbf{y}_i]_{i=1}^N$, $M \times N$ matrix of data samples.
K, desired number of clusters.
Initialization
- Assign $d = R - 1$.
- Initialize the $M \times d$ matrix $\mathbf{W}$ using PCA.

Algorithm
Loop until convergence
- Compute the $d-$dimensional data samples $\mathbf{X} = \mathbf{W}^T\mathbf{Y}$.
- Perform K-lines clustering on $\mathbf{X}$ and evaluate the membership sets $\{\mathcal{C}_k\}_{k=1}^K$.
- From the memberships $\{\mathcal{C}_k\}_{k=1}^K$, compute the sets $\{\mathcal{M}_l\}_{l=1}^{2K}$, using the procedure described in Section 4.4.1.
- Compute each cluster center $\boldsymbol{\psi}_k$ in $\boldsymbol{\Psi}$ as left singular vector corresponding to the largest singular value in the SVD of $\mathbf{Y}_k = [\mathbf{y}_n]_{n \in \mathcal{C}_k}$.
- Using $\boldsymbol{\Psi}$, obtain the columns in $\boldsymbol{\Omega}$ from (4.12).
- Evaluate the between-class and within-class similarity measures, $\mathbf{D}_w$ and $\mathbf{D}_b$ using (4.14) and (4.15) respectively.
- Update $\mathbf{W}$ using the eigenvectors corresponding to the d largest eigenvalues of $\mathbf{D}_b^{-1}\mathbf{D}_w$.
end

analysis proposed in the previous section, in order to evaluate the cluster memberships and the linear discriminant $\mathbf{W}$. As the first step, we perform K-lines clustering on the projected data $\mathbf{X} = \mathbf{W}^T\mathbf{Y}$ in $\mathbb{R}^d$ space. The number of reduced of dimensions is set at $R - 1$ where R is the number of columns of the mixing matrix, whereas the number of clusters is set at $K > R$, i.e., we typically overestimate the number of clusters. The cluster memberships computed in the $\mathbb{R}^d$ space are given by $\{\mathcal{M}_l\}_{l=1}^{2K}$. The memberships are carried over from $\mathbb{R}^d$ space to $\mathbb{R}^M$ space. Using the relation between the membership sets $\{\mathcal{M}_l\}_{l=1}^{2K}$ and $\{\mathcal{C}_k\}_{k=1}^K$, the cluster centers $\boldsymbol{\Psi}$ and hence $\boldsymbol{\Omega}$ are computed. The cluster center $\boldsymbol{\psi}_k$ corresponding to a membership set $\mathcal{C}_k$ will be the left singular vector corresponding to the largest singular value of the training vectors belonging to the set. The discriminant $\mathbf{W}$ is computed using the similarity measures

$\mathbf{D}_w$ and $\mathbf{D}_b$ that are obtained using the updated $\mathbf{\Omega}$. The Khyp-LDA algorithm proceeds until the memberships do not change over subsequent iterations. Finally, out of the K clusters, only the most significant R clusters are chosen as the columns of the mixing matrix based on their confidence indices as described in Section 4.6.1.

4.5 Mixing Matrix Estimation in Underdetermined BSS

The Khyp-LDA algorithm cannot be used to estimate $\mathbf{G}$ in underdetermined mixing conditions, since it projects the data onto $R-1$ dimensions for computing the cluster memberships and in underdetermined conditions we have $R > M$, where M is the ambient dimension of the data. This can be overcome by first projecting the data onto a feature space $\mathcal{F}$ whose dimensions are much higher than M, prior to performing discriminative clustering. Since $\mathcal{F}$ is equipped with a reproducing kernel, both linear discriminant analysis and K-lines clustering can be performed using the *kernel trick*. The generalized discriminant analysis (GDA) proposed in [136] performs discriminant analysis in the feature space using kernel similarity matrices and elementary matrix techniques. In this section, we begin by extending the discriminant analysis procedure, performed using similarity measures, described in Section 4.4.1 to the feature space and develop the modified GDA algorithm. We then propose the Khyp-GDA algorithm that performs K-lines clustering using the data projected onto the subspace of $\mathcal{F}$ identified by GDA.

4.5.1 GDA using Similarity Measures

The cluster centroids $\mathbf{\Psi}$ and $\mathbf{\Omega}$ can be defined in the feature space as $\Phi(\mathbf{\Psi})$ and $\Phi(\mathbf{\Omega})$. We denote the overall cluster center $\boldsymbol{\omega}$ in the feature space by $\Phi(\boldsymbol{\omega})$. Let $\mathbf{N}_{\mathbf{\Psi}}$, $\mathbf{N}_{\mathbf{\Omega}}$ and $N_{\boldsymbol{\omega}}$ be the matrices that normalize the cluster centroids $\Phi(\mathbf{\Psi})$, $\Phi(\mathbf{\Omega})$ and $\Phi(\boldsymbol{\omega})$ respectively. They are computed similar to the normalization matrix in (3.49). The within-class and between-class similarity measures are computed similar to (4.18) and (4.19), although in the feature space. The non-symmetric within-class similarity is

obtained by performing the feature space substitutions $\Phi(\mathbf{Y})$ and $\Phi(\mathbf{\Omega})\mathbf{N}_\Omega$ in place of $\mathbf{Y}$ and $\mathbf{\Omega}$ in the first RHS term of (4.18) and is given as

$$\mathbf{V}_u = \Phi(\mathbf{Y})\mathbf{M}(\Phi(\mathbf{\Omega})\mathbf{N}_{\mathbf{\Omega}})^T = \Phi(\mathbf{Y})\mathbf{M}\mathbf{N}_{\mathbf{\Omega}}^T(\Phi(\mathbf{Y})\mathbf{E})^T \tag{4.20}$$

$$= \Phi(\mathbf{Y})\mathbf{T}_1\Phi(\mathbf{Y})^T, \tag{4.21}$$

where $\mathbf{E} = \mathbf{M} \odot \boldsymbol{\gamma}$ and $\boldsymbol{\gamma} = \Phi(\mathbf{Y})^T\Phi(\mathbf{\Omega})$. The non-symmetric between-class similarity is now obtained using feature space substitutions in (4.19) as

$$\mathbf{B}_u = \Phi(\mathbf{\Psi})\mathbf{N}_{\mathbf{\Psi}}\mathbf{M}_p(\Phi(\boldsymbol{\omega}_p)N_{\boldsymbol{\omega}})^T + \Phi(\mathbf{\Psi})\mathbf{N}_{\mathbf{\Psi}}\mathbf{M}_m(\Phi(\boldsymbol{\omega}_m)N_{\boldsymbol{\omega}})^T, \tag{4.22}$$

$$= \Phi(\mathbf{Y})\mathbf{H}\mathbf{N}_{\mathbf{\Psi}}\mathbf{M}_pN_{\boldsymbol{\omega}}(\Phi(\mathbf{Y})\mathbf{E}_p)^T + \Phi(\mathbf{Y})\mathbf{H}\mathbf{N}_{\mathbf{\Psi}}\mathbf{M}_mN_{\boldsymbol{\omega}}(\Phi(\mathbf{Y})\mathbf{E}_m)^T,$$

$$= \Phi(\mathbf{Y})\mathbf{T}_2\Phi(\mathbf{Y})^T, \tag{4.23}$$

where

$$\mathbf{E}_p = \mathbf{M}_p \odot \boldsymbol{\alpha}_p, \qquad \boldsymbol{\alpha}_p = \Phi(\mathbf{\Psi})^T\Phi(\boldsymbol{\omega}_p) = \mathbf{N}_{\mathbf{\Psi}}^T\mathbf{H}^T\Phi(\mathbf{Y})^T\Phi(\mathbf{Y})\hat{\boldsymbol{\alpha}}_p, \tag{4.24}$$

$$\mathbf{E}_m = \mathbf{M}_m \odot \boldsymbol{\alpha}_m, \text{ and } \boldsymbol{\alpha}_m = \Phi(\mathbf{\Psi})^T\Phi(\boldsymbol{\omega}_m) = \mathbf{N}_{\mathbf{\Psi}}^T\mathbf{H}^T\Phi(\mathbf{Y})^T\Phi(\mathbf{Y})\hat{\boldsymbol{\alpha}}_m. \tag{4.25}$$

When computing (4.24) and (4.25), we express $\Phi(\mathbf{\Psi}) = \Phi(\mathbf{Y})\mathbf{H}\mathbf{N}_{\mathbf{\Psi}}$, $\Phi(\boldsymbol{\omega}_p) = \Phi(\mathbf{Y})\hat{\boldsymbol{\alpha}}_p$ and $\Phi(\boldsymbol{\omega}_m) = \Phi(\mathbf{Y})\hat{\boldsymbol{\alpha}}_m$, where $\hat{\boldsymbol{\alpha}}_p$ and $\hat{\boldsymbol{\alpha}}_m$ are the weights obtained when computing the left singular vector of $\Phi(\mathbf{Y})$ iteratively, and $\mathbf{H}$ is the matrix corresponding to the membership sets $\{\mathcal{C}_k\}$. In order to compute the discriminant directions in the feature space, we perform GDA using the symmetric within- and between-class similarity matrices

$$\mathbf{V} = \mathbf{V}_u + \mathbf{V}_u^T, \tag{4.26}$$

$$\mathbf{B} = \mathbf{B}_u + \mathbf{B}_u^T. \tag{4.27}$$

Denoting $\mathbf{U} = [\mathbf{u}_1\ \mathbf{u}_2\ \ldots\ \mathbf{u}_d]$ as set of directions of maximal discrimination in $\mathcal{F}$, any discriminant direction $\mathbf{u}$ can be expressed as

$$\lambda\mathbf{B}\mathbf{u} = \mathbf{V}\mathbf{u}. \tag{4.28}$$

Since, every vector from $\mathbf{U}$ lies in the span of $\Phi(\mathbf{Y})$,

$$\mathbf{u} = \Phi(\mathbf{Y})\boldsymbol{\delta}. \tag{4.29}$$

Now, we premultiply $\Phi(\mathbf{Y})^T$ on both sides of (4.28)

$$\lambda\Phi(\mathbf{Y})^T\mathbf{B}\mathbf{u} = \Phi(\mathbf{Y})^T\mathbf{V}\mathbf{u}, \tag{4.30}$$

From (4.21), (4.26), and (4.29), we can write

$$\begin{aligned}\Phi(\mathbf{Y})^T\mathbf{V}\mathbf{u} &= \Phi(\mathbf{Y})^T\Phi(\mathbf{Y})(\mathbf{T}_1 + \mathbf{T}_1^T)\Phi(\mathbf{Y})^T\Phi(\mathbf{Y})\boldsymbol{\delta}, \\ &= \mathbf{K}(\mathbf{T}_1 + \mathbf{T}_1^T)\mathbf{K}\boldsymbol{\delta}. \end{aligned} \tag{4.31}$$

Similarly, using (4.23), (4.27), and (4.29), we obtain

$$\lambda\Phi(\mathbf{Y})^T\mathbf{B}\mathbf{u} = \lambda\mathbf{K}(\mathbf{T}_2 + \mathbf{T}_2^T)\mathbf{K}\boldsymbol{\delta}.$$

Hence, we can compute λ in (4.30) as

$$\lambda = \frac{\boldsymbol{\delta}^T\mathbf{K}(\mathbf{T}_1 + \mathbf{T}_1^T)\mathbf{K}\boldsymbol{\delta}}{\boldsymbol{\delta}^T\mathbf{K}(\mathbf{T}_2 + \mathbf{T}_2^T)\mathbf{K}\boldsymbol{\delta}}. \tag{4.32}$$

We employ an approach similar to the one used in [136] and perform the eigen decomposition of $\mathbf{K} = \mathbf{P}\mathbf{F}\mathbf{P}^T$. Substituting the eigen decomposition of $\mathbf{K}$ in (4.32) and assigning $\boldsymbol{\beta} = \mathbf{F}\mathbf{P}^T\boldsymbol{\delta}$, we obtain

$$\lambda = \frac{\boldsymbol{\beta}^T\mathbf{P}^T(\mathbf{T}_1 + \mathbf{T}_1^T)\mathbf{P}\boldsymbol{\beta}}{\boldsymbol{\beta}^T\mathbf{P}^T(\mathbf{T}_2 + \mathbf{T}_2^T)\mathbf{P}\boldsymbol{\beta}}. \tag{4.33}$$

This is equivalent to solving [136, App. A]

$$\lambda\mathbf{P}^T(\mathbf{T}_1 + \mathbf{T}_1^T)\mathbf{P}\boldsymbol{\beta} = \mathbf{P}^T(\mathbf{T}_2 + \mathbf{T}_2^T)\mathbf{P}\boldsymbol{\beta}, \tag{4.34}$$

where $\boldsymbol{\beta}$ is obtained as a generalized eigenvector of the system. Generalized eigen decomposition does not have a closed-form expression, but can be robustly obtained using numerical techniques [138]. Note that $\boldsymbol{\beta}$ is a representative variable for a generalized eigenvector and multiple eigenvectors, $[\boldsymbol{\beta}_i]_{i=1}^d$, corresponding to the d largest

Table 4.2: Khyp-GDA clustering for mixing matrix estimation in underdetermined BSS.

Input
$\mathbf{Y} = [\mathbf{y}_i]_{i=1}^N$, $M \times N$ matrix of data samples.
$\mathbf{K}$, $N \times N$ kernel matrix.
K, desired number of clusters.
Initialization
- Assign $d = R - 1$.
- Initialize the cluster membership matrix $\mathbf{Z}$ and $\mathbf{H}$ using Kernel K-lines clustering (Section 3.6.3).
- Compute $\mathbf{E}_p$ and $\mathbf{E}_m$ using (4.24) and (4.25).
Loop until convergence
  GDA step:
  - Perform eigen decomposition of $\mathbf{K} = \mathbf{PFP}^T$.
  - Evaluate $\mathbf{T}_1 = \mathbf{MN}_{\mathbf{\Omega}}^T\mathbf{E}^T$ and $\mathbf{T}_2 = \mathbf{HN}_{\mathbf{\Psi}}\left(\mathbf{M}_p\mathbf{E}_p^T + \mathbf{M}_m\mathbf{E}_m^T\right)N_{\boldsymbol{\omega}}$.
  - Compute the d generalized eigenvectors $[\boldsymbol{\beta}_i]_{i=1}^d$ (see (4.34) and the description provided therein).
  - Evaluate $\boldsymbol{\delta}_i = \mathbf{PF}^{-1}\boldsymbol{\beta}_i$ and divide by $\sqrt{\boldsymbol{\delta}_i^T\mathbf{K}\boldsymbol{\delta}_i}$ to ensure that $\mathbf{u}_i^T\mathbf{u}_i = 1$.
  Kernel K-lines clustering step:
  Loop until convergence
    Loop for L iterations
      - Compute $\boldsymbol{\alpha}$ using (4.39).
      - Compute $\mathbf{H} = \mathbf{Z} \odot \boldsymbol{\alpha}$.
    end
    - Update $\mathbf{Z}$ by identifying the index of absolute maximum in each row of $\boldsymbol{\alpha}$.
  end
end

generalized eigenvalues can also be obtained using the generalized eigen decomposition. $\boldsymbol{\delta}$ will be obtained from $\boldsymbol{\beta}$ as $\boldsymbol{\delta} = \mathbf{PF}^{-1}\boldsymbol{\beta}$. In order to ensure that $\mathbf{u}^T\mathbf{u} = 1$, $\boldsymbol{\delta}$ is normalized by dividing it using $\sqrt{\boldsymbol{\delta}^T\mathbf{K}\boldsymbol{\delta}}$, according to (4.29). Now, we can project any vector $\Phi(\mathbf{r})$ in $\mathcal{F}$ onto the direction $\mathbf{u}$ as

$$\mathbf{u}^T\Phi(\mathbf{r}) = \boldsymbol{\delta}^T\mathbf{K}(\mathbf{Y}, \mathbf{r}). \tag{4.35}$$

4.5.2 Khyp-GDA Algorithm

Similar to the Khyp-LDA algorithm, we identify the directions of maximum discrimination $\mathbf{U}$ and project $\Phi(\mathbf{Y})$ to the subspace $\mathcal{F}_{\mathbf{U}}$ identified by $\mathbf{U}$. Kernel K-lines

clustering will be performed on $\mathbf{U}^T\Phi(\mathbf{Y})$. Table 4.2 shows the steps involved in the Khyp-GDA algorithm. We begin by initializing the cluster membership matrix $\mathbf{Z}$ and computing the kernel matrix $\mathbf{K}$. Using the class membership, we perform GDA and compute $\boldsymbol{\Delta} = [\boldsymbol{\delta}_1\ \boldsymbol{\delta}_2 \dots \boldsymbol{\delta}_d]$. The next step is to project the feature space data onto the subspace $\mathcal{F}_{\mathbf{U}}$ as

$$\mathbf{U}^T\Phi(\mathbf{Y}) = \boldsymbol{\Delta}^T\mathbf{K}. \tag{4.36}$$

The normalized cluster centroids after projection can be expressed as

$$\Phi(\hat{\boldsymbol{\Psi}}) = \mathbf{U}^T\Phi(\boldsymbol{\Psi})\mathbf{N}_{\hat{\boldsymbol{\Psi}}} = \mathbf{U}^T\Phi(\mathbf{Y})\mathbf{H}\mathbf{N}_{\hat{\boldsymbol{\Psi}}}, \tag{4.37}$$

$$= \boldsymbol{\Delta}^T\mathbf{K}\mathbf{H}\mathbf{N}_{\hat{\boldsymbol{\Psi}}}. \tag{4.38}$$

where $\mathbf{N}_{\hat{\boldsymbol{\Psi}}} = \left[diag(\mathbf{H}^T\mathbf{K}^T\boldsymbol{\Delta}\boldsymbol{\Delta}^T\mathbf{K}\mathbf{H})\right]^{-1/2}$. The coefficients $\boldsymbol{\alpha}$ in kernel K-lines clustering are computed as

$$\boldsymbol{\alpha} = (\mathbf{U}^T\Phi(\mathbf{Y}))^T\Phi(\hat{\boldsymbol{\Psi}}) = \mathbf{K}^T\boldsymbol{\Delta}\boldsymbol{\Delta}^T\mathbf{K}\mathbf{H}\mathbf{N}_{\hat{\boldsymbol{\Psi}}}. \tag{4.39}$$

In the algorithm described in Table 4.2, the outer loop of the kernel K-lines clustering step iteratively updates the clustering and the inner loop computes the coefficients.

4.6 Simulation Results

In this section, we study the behavior of the proposed clustering approaches under different levels of sparsity and disjoint orthogonality [126] in the sources. Furthermore, we employ the proposed algorithms for clustering SSPs obtained from speech mixtures, as described in Section 4.3.1, and demonstrate performance improvements in mixing matrix estimation. In all the simulations presented, the performance of estimation is evaluated using signal to interference ratio (SIR) between the original mixing matrix $\mathbf{G}$ and its estimate $\hat{\mathbf{G}}$,

$$SIR(\mathbf{G}, \hat{\mathbf{G}}) = 20\log_{10}\left(\frac{\|\mathbf{G}\|_F}{\|\mathbf{G} - \hat{\mathbf{G}}\|_F}\right), \tag{4.40}$$

where $\|.\|_F$ denotes the Frobenius norm of the argument matrix.

4.6.1 Synthetic Data

The simulations in this section are performed using randomly generated source and mixing matrices. We demonstrate the effect of sparsity in the sources and the level of disjoint orthogonality in the source matrix, on mixing matrix estimation using the proposed clustering algorithms.

Effect of Sparsity on Estimation Performance

In this experiment, five source signals of 1000 samples were realized from a uniform distribution in $[-2, 2]$ and made β-sparse by forcing only β randomly chosen samples to be non-zero. The signals were stacked in the source matrix $\mathbf{S} \in \mathbb{R}^{5\times 1000}$. The sparsity level was varied across different values in the range 100 to 350. The number of sensors P was fixed at 3 and 7 for the underdetermined and the overdetermined mixing scenarios respectively. In both cases, the entries of the mixing matrix $\mathbf{G} \in \mathbb{R}^{P\times 5}$ were realized from a zero mean, unit variance Gaussian distribution and each column of $\mathbf{G}$ was rescaled to unit norm. As described earlier, the Khyp-GDA algorithm was used for underdetermined mixing matrix estimation and the Khyp-LDA algorithm was employed for the overdetermined estimation.

In both the mixing conditions, we followed the approach described in [23], where the number of cluster centers was overestimated and the outliers were rejected based on a confidence index for each cluster center. Though the original number of sources was $R = 5$, we chose $K = 25$. Note that we fixed $d = 4$ for the Khyp-LDA (Table 4.1) and the Khyp-GDA (Table 4.2) algorithms based on the original number of sources, and not on the overestimated number of clusters. The singular value corresponding to each cluster center was used as its confidence measure and they were sorted to identify the dominant directions. For Khyp-GDA, we employed the RBF kernel defined in (3.47) to perform the feature space operations. Figures 4.2(a) and 4.2(b) show the SIR of the mixing matrix estimation for source signals

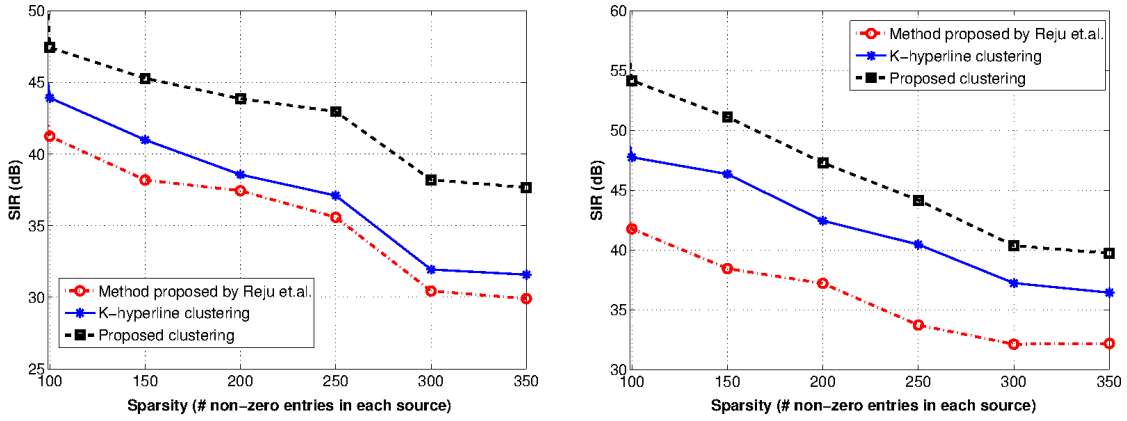


Figure 4.2: Mixing matrix estimation performance for different levels of sparsity in the source signals: (a) Overdetermined case (number of sources $R = 5$, number of sensors $P = 7$), (b) Underdetermined case ($R = 5, P = 3$).

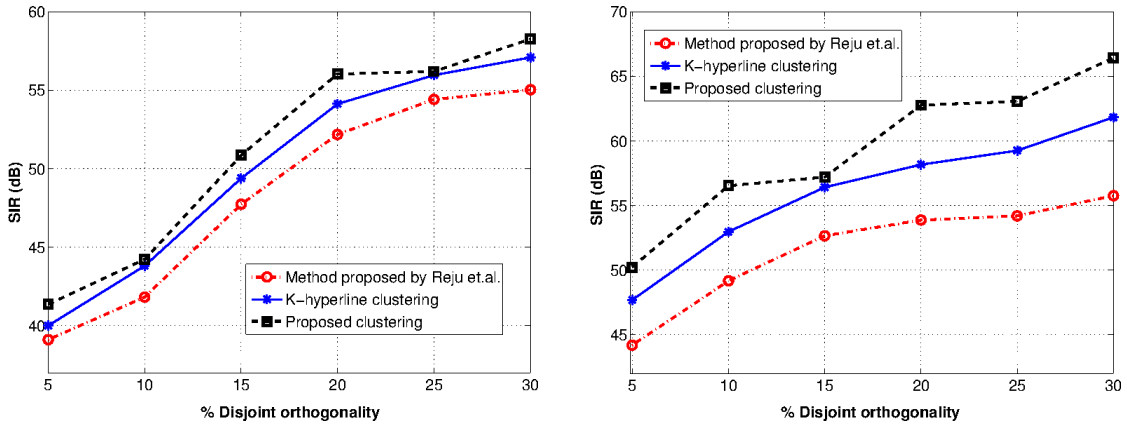


Figure 4.3: Mixing matrix estimation performance for different levels of disjoint orthogonality between the sources: (a) Overdetermined case ($R = 5, P = 7$), (b) Underdetermined case ($R = 5, P = 3$).

with different levels of sparsity. The results obtained were averaged over 50 iterations and for each iteration a different mixing matrix $\mathbf{G}$ was randomly generated. In addition to the proposed approaches, we performed estimation using the hierarchical clustering approach proposed in [3] and the K-lines clustering proposed in [23]. As it can be easily observed, the proposed algorithms outperformed the other clustering procedures in all cases of sparsity.

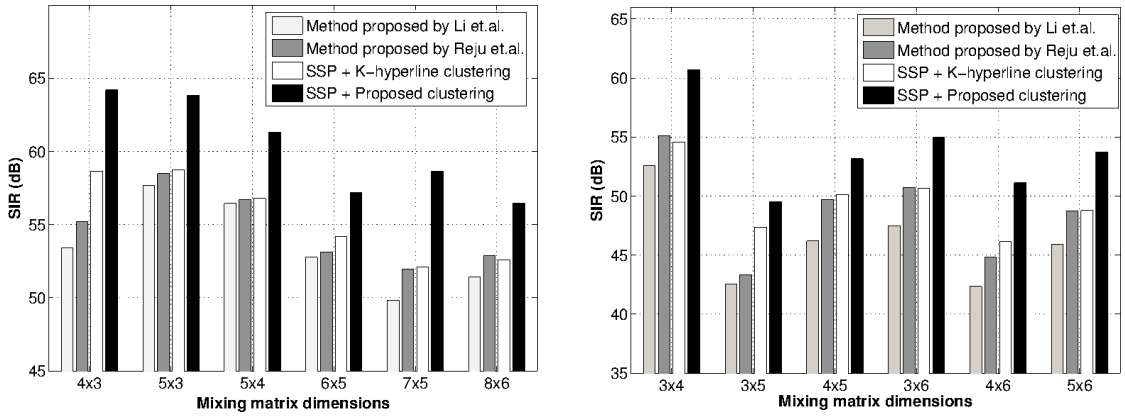
Effect of Disjoint Orthogonality on Estimation Performance

In order to study the effect of disjoint orthogonality among the sources on the estimation performance, we generated a random dataset using an approach similar to [129]. The entries in $\mathbf{S} \in \mathbb{R}^{5\times 10000}$, were realized from a uniform distribution in the range $[-5, 5]$. We performed simulations with 5% to 25% of samples in the source matrix satisfying the disjoint orthogonality condition. For example, in the case of 10% disjoint orthogonality, we chose 5 disjoint sets of 200 column indices in the source matrix and made only one entry in each of those columns to be non-zero. The i^{th} set of 200 columns had their i^{th} entry to be non-zero ($i = 1, \cdots, 5$). Similar to the previous experiment, the number of sensors were fixed at 3 and 7 respectively. The mixing matrix $\mathbf{G}$ was randomly generated as described in Section 4.6.1 and the results were averaged over 50 iterations. In this case, the number of cluster centers was overestimated as $K = 250$. The improvement in the estimation performance obtained using the proposed discriminative clustering approaches when compared to the K-lines [23] and the hierarchical clustering [3] methods is evident in both the cases, as seen in Figures 4.3(a) and 4.3(b).

4.6.2 Mixing Matrix Estimation for Speech Mixtures

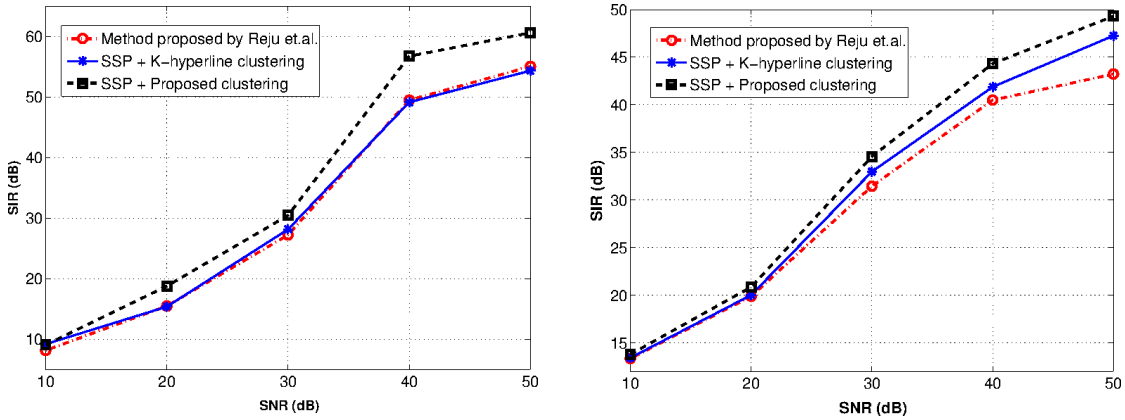
In this section, we describe the results obtained using the proposed algorithms to cluster the SSPs from the TF representation of speech mixtures. The dataset used for the experiments contained a set of 6 speech utterances of 10 seconds each, sampled at 16 kHz [3]. We compared the estimation performances obtained using: (a) Hierarchical clustering of the SSPs [3], (b) K-lines clustering [23] of the SSPs, (c) Proposed discriminative clustering of the SSPs and (d) Time-frequency ratio based method proposed in [129].

We performed multiple simulations with different configurations of R and P and we report the results obtained by averaging 100 trials. Mixing matrix estimation



(a) (b)

Figure 4.4: Performance of mixing matrix estimation from speech mixtures, at different configurations of R and P: (a) Overdetermined BSS, (b) Underdetermined BSS.



(a) (b)

Figure 4.5: Performance of mixing matrix estimation from speech mixtures, under different observation noise conditions: (a) Overdetermined case ($R = 3, P = 4$), (b) Underdetermined case ($R = 5, P = 3$).

was performed in the STFT domain and the same number of frequency bins was used for all the algorithms. For each speech signal, the STFT was obtained using a Hann window of 1024 samples with 12.5% overlap. For the methods (a)-(c), we identify the SSPs by fixing $\Delta\theta$ and we sorted the frequency bins in descending order of their variance such that very few frequency bins can be used to estimate the mixing matrix [3]. In our experiments, we observed that for an appropriate choice of $\Delta\theta$,

a good estimate of the mixing matrix is obtained using only 10% of the bins. Both the real and imaginary components of the SSPs were stacked together to perform the clustering. For the method (d), we used the parameters specified in [129] and the hierarchical clustering method used in [3] to estimate $\mathbf{G}$.

Figure 4.4 illustrates the estimation performance (SIR) of the four methods in different underdetermined and overdetermined mixing conditions. The parameter $\Delta\theta$ was fixed at 0.8 degrees and the SSPs were identified using the top 10% of the frequency bins, sorted by decreasing variance. As it can be observed, the proposed algorithms outperformed the other baseline methods in all cases of mixing. The effect of observation noise on the estimation was also analyzed by fixing the target SNR between 10 dB and 50 dB and evaluating the performance. Under noisy conditions, we tested different values of $\Delta\theta$, between 0.8 and 10 degrees, for identifying the SSPs and chose the one that resulted in the best SIR. Figure 4.5 shows the SIR obtained at different noise levels for the cases with $\{R = 3, P = 4\}$ and $\{R = 5, P = 3\}$. This simulation clearly demonstrates the performance improvement obtained by using the proposed algorithms under noisy conditions.

Chapter 5

MULTILEVEL DICTIONARY LEARNING FOR SPARSE REPRESENTATIONS

5.1 Problem Statement

The goal is to construct stable and generalizable dictionaries for sparse approximation of natural image patches. A simple example of learning a dictionary with two levels is demonstrated in Figure 5.1. The properties and performance of this learning algorithm will be analyzed in detail in this chapter. The multilevel dictionary (MLD) learning algorithm is a hierarchical procedure where the dictionary atoms in each level are obtained using a 1-D subspace clustering algorithm, which we refer to as *K-lines clustering* [23]. Note that in the papers [23, 35, 39] the procedure has been referred to as K-hyperline clustering. But in this chapter, we prefer to use the term K-lines clustering, since a 1-D subspace in any number of dimensions is referred only to as a line, and not as a hyperline. The proposed algorithm builds global dictionaries using a set of randomly chosen training patches obtained from a large collection of natural images that can generalize well to any test set of patches. For a learned dictionary to provide a good approximation, the test data must be similar to the data samples used for training. Since local regions of natural images have high redundancy and consistent statistical properties [7], learning global dictionaries from a random collection of natural image patches will provide a good representation for patches from images not in the training set. The effectiveness of such dictionaries have been demonstrated in denoising [139] and compressed recovery [140].

A learning algorithm is a map from the space of training examples to the hypothesis space of functional solutions. Algorithmic stability characterizes the behavior of a learning algorithm with respect to the perturbations of its training set [108], and generalization ensures that the expected error of the learned function with respect to the novel test data will be close to the average empirical training error [141]. In clustering, the learned function is completely characterized by the cluster centers.

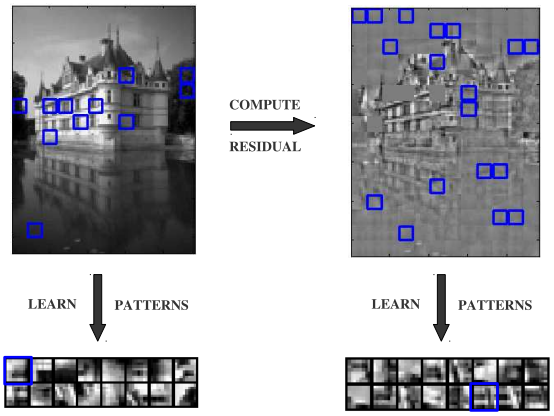


Figure 5.1: Features learned at two levels from non-overlapping patches (8×8) of a 128×96 image. In each level, the patches that are highlighted in the image share similar information and hence jointly correspond to a learned pattern (also highlighted).

Stability of a clustering algorithm implies that the cluster centroids learned by the algorithm are not significantly different when different sets of i.i.d. samples from the same probability space are used for training [26]. When there is a unique minimizer to the clustering objective with respect to the underlying data distribution, stability of a clustering algorithm is guaranteed [24] and this analysis has been extended to characterize the stability of K-means clustering in terms of the number of minimizers [25]. In [35], the stability properties of the K-lines clustering algorithm have been analyzed and they have been shown to be similar to those of K-means clustering. Note that all the stability characterizations depend only on the underlying data distribution and the number of clusters, and not on the actual training data itself. Generalization implies that the average empirical training error becomes asymptotically close to the expected error with respect to the probability space of data, as the number of training samples $T \rightarrow \infty$. In [142], the generalization bound for sparse coding in terms of the number of samples T, also referred to as sample complexity, is derived and in [143] the bound is improved by assuming a class of dictionaries that are nearly orthogonal.

The algorithmic stability of dictionary learning methods has not been discussed in the literature until now, to the best of our knowledge. Given a sufficiently

large training set, a stable learning algorithm will result in global dictionaries that will depend only on the probability space to which the training samples belong and not on the actual samples themselves. Generalization ensures that such global dictionaries learned result in a good performance with test data. In other words, the asymptotic stability and generalization of a dictionary learning algorithm provides the theoretical justification for the uniformly good performance of global dictionaries learned from an arbitrary training set. We study the stability properties of the proposed MLD learning algorithm and prove that it is asymptotically stable. We also show that the proposed algorithm generalizes asymptotically.

5.2 Multilevel Dictionary Learning

In this section, we motivate and develop a multilevel dictionary learning approach for sparse representations, whose algorithmic stability and generalizability will be proved in Section 5.4. Furthermore, we propose the RM-OMP algorithm, that can be used to obtain sparse codes for a test image using the multilevel dictionary.

5.2.1 Motivation for Multilevel Learning

Our motivation for learning an MLD is two-fold. Firstly we require a global dictionary that can exploit, (a) the redundancy observed across local regions in natural images and, (b) the hierarchy of patterns found in training image patches. Secondly, the learning procedure must be provably stable, with respect to the notion of algorithmic stability, and generalizable.

The generative model for sparse coding is well suited for natural signals and images as they can be represented using a sparse linear combination of elementary features chosen from a dictionary [18]. The redundancy in the local regions of natural images [7] allows for the design of global dictionaries that can generalize well to a wide range of images. Global dictionaries learned from a set of randomly chosen patches from natural images have been successfully used for denoising [139], com-

pressed sensing [140] and classification [98]. In addition to exhibiting redundancy, the natural image patches typically contain either geometric patterns or stochastic textures or a combination of both. This fact is demonstrated in [144], where the authors define two types of atomic subspaces to model image patches: subspaces of low dimensions (explicit manifolds) for primitive geometric patterns and subspaces of high dimensions (implicit manifolds) for stochastic textures. Since the image patches can contain both geometric and stochastic structures, a hybrid combination of explicit and implicit manifolds can be used for modeling them [144]. The proposed MLD algorithm learns global representative patterns in multiple levels, according to the order of their energy contribution. Since the geometric patterns usually are of higher energy when compared to stochastic textures in images, geometric patterns are learned in the first few levels and stochastic textures are learned in the last few levels.

Considering the dictionary learning formulation, it can be seen that clustering algorithms such as the K-means and the K-lines can be obtained by constraining the desired sparsity to be 1. Since the stability characteristics of clustering algorithms are well understood, employing similar tools to analyze the more general dictionary learning can be beneficial. Note that the proposed algorithm poses dictionary learning as performing K-lines clustering in multiple levels and hence in this case we can use the stability characteristics of the clustering algorithm to study the stability of multilevel learning. Furthermore, by exploiting the fact that the distortion function class for each level of learning is uniform Donsker, the generalizability of the algorithm can also be proved. Note that multilevel learning is different from the work in [145], where multiple sub-dictionaries are designed and one of them is chosen for representing a group of patches.

5.2.2 Proposed MLD Learning Algorithm

We denote the MLD as $\mathbf{\Psi} = [\mathbf{\Psi}_1 \mathbf{\Psi}_2 ... \mathbf{\Psi}_L]$, and the coefficient matrix as $\mathbf{A} = [\mathbf{A}_1^T \mathbf{A}_2^T ... \mathbf{A}_L^T]^T$. Here, $\mathbf{\Psi}_l$ is the sub-dictionary and $\mathbf{A}_l$ is the coefficient matrix for level l. The approximation in level l is expressed as

$$\mathbf{R}_{l-1} = \mathbf{\Psi}_l \mathbf{A}_l + \mathbf{R}_l, \text{ for } l = 1, ..., L, \tag{5.1}$$

where $\mathbf{R}_{l-1}$, $\mathbf{R}_l$ are the residuals for the levels $l-1$ and l respectively and $\mathbf{R}_0 = \mathbf{Y}$, the matrix of training image patches. This implies that the residual matrix in level $l-1$ serves as the training data for level l. Note that the sparsity of the representation in each level is fixed at 1. Hence, the overall approximation for all levels is

$$\mathbf{Y} = \sum_{l=1}^{L} \mathbf{\Psi}_l \mathbf{A}_l + \mathbf{R}_L. \tag{5.2}$$

MLD learning can be interpreted as a block-based dictionary learning problem with unit sparsity per block, where the sub-dictionary in each block can allow only a 1-sparse representation and each block corresponds to a level. The sub-dictionary for level l, $\mathbf{\Psi}_l$, is the set of cluster centroids learned from the training matrix for that level, $\mathbf{R}_{l-1}$, using K-lines clustering. MLD learning can be formally stated as an optimization problem that proceeds from the first level until the stopping criteria is reached. For level l, the optimization problem is

$$\begin{aligned} \underset{\mathbf{\Psi}_l, \mathbf{A}_l}{\operatorname{argmin}} \|\mathbf{R}_{l-1} - \mathbf{\Psi}_l \mathbf{A}_l\|_F^2 \text{ subject to } \|\mathbf{a}_{l,i}\|_0 \leq 1, \\ \text{for } i = \{1, ..., T\}, \end{aligned} \tag{5.3}$$

along with the constraint that the columns of $\mathbf{\Psi}_l$ have unit ℓ_2 norm, where $\mathbf{a}_{l,i}$ is the i^{th} column of $\mathbf{A}_l$ and T is the number of columns in $\mathbf{A}_l$. We adopt the notation $\{\mathbf{\Psi}_l, \mathbf{A}_l\} = \text{KLC}(\mathbf{R}_{l-1}, K_l)$ to denote the problem in (5.3) where K_l is the number of atoms in the sub-dictionary $\mathbf{\Psi}_l$. The stopping criteria is provided either by imposing

Table 5.1: Algorithm for building a multilevel dictionary.

Input
$\mathbf{Y} = [\mathbf{y}_i]_{i=1}^{T}$, $M \times T$ matrix of training vectors.
L, maximum number of levels of the dictionary.
K_l, number of dictionary elements in level l, $l = \{1, 2, ..., L\}$.
ϵ, error goal of the representation.

Output
$\mathbf{\Psi}_l$, adapted sub-dictionary for level l.

Algorithm
Initialize: $l = 1$ and $\mathbf{R}_0 = \mathbf{Y}$.
$\Lambda_0 = \{i \mid \|\mathbf{r}_{0,i}\|_2^2 > \epsilon, 1 \leq i \leq T\}$, index of training vectors with squared norm greater than error goal.
$\hat{\mathbf{R}}_0 = [\mathbf{r}_{0,i}]_{i \in \Lambda_0}$.

while $\Lambda_{l-1} \neq \emptyset$ and $l \leq L$
  Initialize:
    $\mathbf{A}_l$, coefficient matrix, size $K_l \times M$, all zeros.
    $\mathbf{R}_l$, residual matrix for level l, size $M \times T$, all zeros.
  $\{\mathbf{\Psi}_l, \hat{\mathbf{A}}_l\} = \text{KLC}(\hat{\mathbf{R}}_{l-1}, K_l)$.
  $\mathbf{R}_l^t = \hat{\mathbf{R}}_{l-1} - \mathbf{\Psi}_l \hat{\mathbf{A}}_l$.
  $\mathbf{r}_{l,i} = \mathbf{r}_{l,j}^t$ where $i = \Lambda_{l-1}(j),\ \forall j = 1, ..., |\Lambda_{l-1}|$.
  $\mathbf{a}_{l,i} = \hat{\mathbf{a}}_{l,j}$ where $i = \Lambda_{l-1}(j),\ \forall j = 1, ..., |\Lambda_{l-1}|$.
  $\Lambda_l = \{i \mid \|\mathbf{r}_{l,i}\|_2^2 > \epsilon, 1 \leq i \leq T\}$.
  $\hat{\mathbf{R}}_l = [\mathbf{r}_{l,i}]_{i \in \Lambda_l}$.
  $l \leftarrow l + 1$.
end

a limit on the residual representation error or the maximum number of levels (L). Note that the total number of levels is the same as the maximum number of non-zero coefficients (sparsity) of the representation. The error constraint can be stated as, $\|\mathbf{r}_{l,i}\|_2^2 \leq \epsilon, \forall i = 1, ..., T$ for some level l, where ϵ is the error goal.

Table 5.1 lists the MLD learning algorithm with sparsity and error constraints. We use the notation $\Lambda_l(j)$ to denote the j^{th} element of the set Λ_l and $\mathbf{r}_{l,i}$ denotes the i^{th} column vector in the matrix $\mathbf{R}_l$. The set Λ_l contains the indices of the residual vectors of level l whose norm is greater than the error goal. The residual vectors

indexed by Λ_l are stacked in the matrix, $\hat{\mathbf{R}}_l$, which in turn serves as the training matrix for the next level, $l+1$. In MLD learning, for a given level l, the residual $\mathbf{r}_{l,i}$ is orthogonal to the representation $\mathbf{\Psi}_l \mathbf{a}_{l,i}$. This implies that

$$\|\mathbf{r}_{l-1,i}\|_2^2 = \|\mathbf{\Psi}_l \mathbf{a}_{l,i}\|_2^2 + \|\mathbf{r}_{l,i}\|_2^2. \tag{5.4}$$

Combining this with the fact that $\mathbf{y}_i = \sum_{l=1}^{L} \mathbf{\Psi}_l \mathbf{a}_{l,i} + \mathbf{r}_{L,i}$, $\mathbf{a}_{l,i}$ is $1-$sparse, and the columns of $\mathbf{\Psi}_l$ are of unit ℓ_2 norm, we obtain the relation

$$\|\mathbf{y}_i\|_2^2 = \sum_{l=1}^{L} \|\mathbf{a}_{l,i}\|_2^2 + \|\mathbf{r}_{L,i}\|_2^2. \tag{5.5}$$

Equation (5.5) states that the energy of any training vector is equal to the sum of squares of its coefficients and the energy of its residual. From (5.4), we also have that,

$$\|\mathbf{R}_{l-1}\|_F^2 = \|\mathbf{\Psi}_l \mathbf{A}_l\|_F^2 + \|\mathbf{R}_l\|_F^2. \tag{5.6}$$

In our implementation of MLD learning, we include an additional step where the residual at each level is orthogonalized to the dictionary atoms chosen so far, and the coefficients are recomputed. Note that this does not affect any other behavior of the algorithm that is discussed in this section.

The training vectors for the first level of the algorithm, $\mathbf{r}_{0,i}$ lie in the ambient $\mathbb{R}^M$ space and the residuals, $\mathbf{r}_{1,i}$, lie in a finite union of $\mathbb{R}^{M-1}$ subspaces. This is because, for each dictionary atom in the first level, its residual lies in an $M-1$ dimensional space orthogonal to it. In the second level, the dictionary atoms can possibly lie anywhere in $\mathbb{R}^M$, and hence the residuals can lie in a finite union of $\mathbb{R}^{M-1}$ and $\mathbb{R}^{M-2}$ dimensional subspaces. Hence, we can generalize that the dictionary atoms for all levels lie in $\mathbb{R}^M$, whereas the training vectors of level $l \geq 2$, lie in finite unions of $\mathbb{R}^{M-1}, \ldots, \mathbb{R}^{M-l+1}$ dimensional subspaces of the $\mathbb{R}^M$ space.

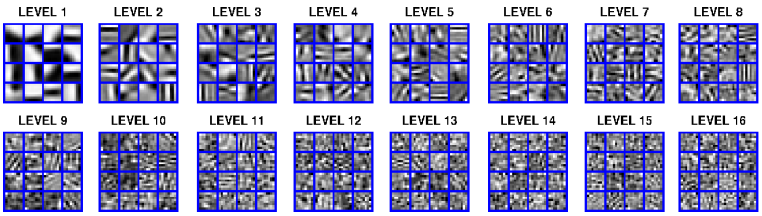


Figure 5.2: Multilevel dictionary, with 16 levels of 16 atoms each, comprises of geometric patterns in the first few levels, stochastic textures in the last few levels and a combination of both in the middle levels.

5.2.3 Convergence

The convergence of MLD learning and the energy hierarchy in the representation obtained using an MLD can be shown by providing two guarantees. The first guarantee is that for a fixed number of atoms per level, the algorithm will converge to the required error within a sufficient number of levels. This is because the K-lines clustering makes the residual energy of the representation smaller than the energy of the training matrix at each level (i.e.) $\|\mathbf{R}_l\|_F^2 < \|\mathbf{R}_{l-1}\|_F^2$. This follows from (5.6) and the fact that $\|\mathbf{\Psi}_l \mathbf{A}_l\|_F^2 > 0$.

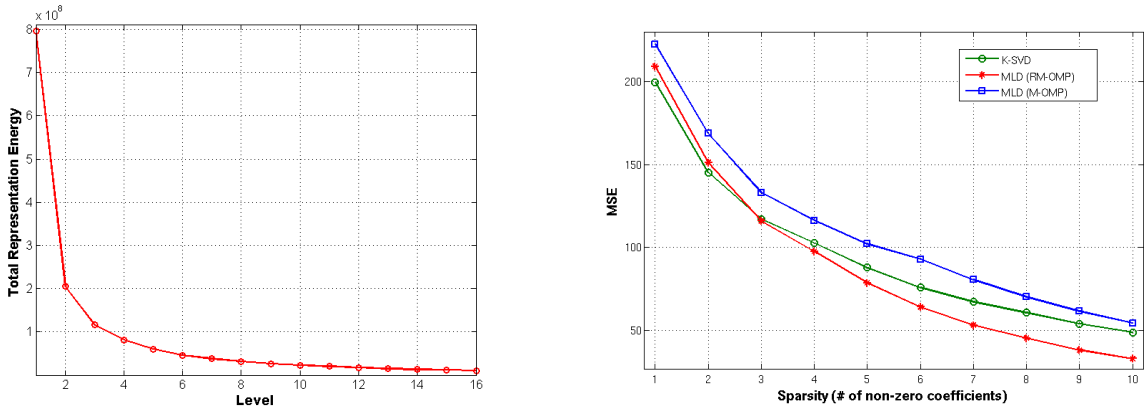
The second guarantee is that for a sufficient number of atoms per level, the representation energy in level $l-1$ will be less than the representation energy in level l. To show this, we first state that for a sufficient number of dictionary atoms per level, $\|\mathbf{\Psi}_l \mathbf{A}_l\|_F^2 > \|\mathbf{R}_l\|_F^2$. This means that for every l

$$\|\mathbf{R}_l\|_F^2 < \|\mathbf{\Psi}_l \mathbf{A}_l\|_F^2 < \|\mathbf{R}_{l-1}\|_F^2, \tag{5.7}$$

because of (5.6). This implies that $\|\mathbf{\Psi}_l \mathbf{A}_l\|_F^2 < \|\mathbf{\Psi}_{l-1} \mathbf{A}_{l-1}\|_F^2$, i.e., the energy of the representation in each level reduces progressively from $l = 1$ to $l = L$.

5.2.4 Sparse Approximation using an MLD

In order to compute sparse codes for novel test data using a multilevel dictionary, we propose to perform reconstruction using a *Multilevel Orthogonal Matching Pursuit* (M-OMP) procedure which evaluates a 1-sparse representation for each level using



(a) (b)

Figure 5.3: (a) Levelwise representation energy for the learned MLD with the BSDS training data set, (b) Comparison of the MSE obtained with the BSDS test dataset using the K-SVD and the MLD dictionaries at different levels of sparsity.

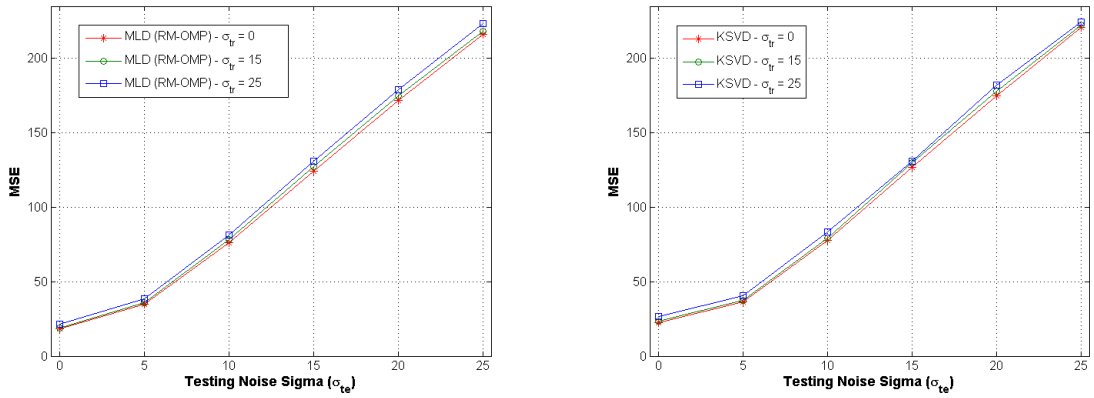
the dictionary atoms from that level, and orthogonalizes the residual to the dictionary atoms chosen so far. Though asymptotic generalization of the M-OMP method will be shown in Section 5.4.5, imposing the energy hierarchy observed in the training process to any test data might result in poor generalization. Hence, there is a need to regularize this procedure such that there is more flexibility in choosing dictionary atoms for representing the test data. Hence, we propose to build a sub-dictionary with atoms selected from the current level as well as the u immediately preceding and following levels, $\tilde{\mathbf{\Phi}}_l = \left[\mathbf{\Phi}_{l-u}\mathbf{\Phi}_{l-(u-1)}\ldots\mathbf{\Phi}_l\ldots\mathbf{\Phi}_{l+(u-1)}\mathbf{\Phi}_{l+u}\right]$, in every step of the pursuit algorithm. In our implementation, we fix $u = 2$ and also reduce the size of the sub-dictionary appropriately when $l \leq u$ and $l > L - u$. The dictionary $\tilde{\mathbf{\Phi}}_l$ is used to compute a 1-sparse representation for that step of the pursuit. It was observed from simulations that the RM-OMP scheme performs better than M-OMP, particularly when the training set is small.

5.2.5 *Dictionaries from the BSDS Dataset*

All simulation results presented in this chapter were obtained with dictionaries learned using randomly chosen patches of size 8×8, extracted from the grayscale images in the training set of the Berkeley segmentation dataset (BSDS) [1]. The number of grayscale patches used for training will be clearly stated for each simulation. As a preprocessing step, the mean value of each training patch was removed. In this section, we will demonstrate the characteristics of MLD learning using an example dictionary learned using $50,000$ patches. Note that, the number of atoms was fixed at 16 per level and the number of levels was fixed at 16, which leads to a total of 256 atoms. For comparison, a global K-SVD dictionary of size 64×256 atoms was learned, with the same training set, using the MATLAB toolbox available online [4]. In this case, the desired sparsity, which refers to the number of non-zero coefficients (S), was fixed at 16. Initial dictionary atoms for the K-SVD algorithm and for each level of MLD learning were obtained using the K-means clustering procedure.

Figure 5.2 illustrates the multilevel dictionary designed using the algorithm in Table 5.1. Note that no noise was added to the image patches during learning. As it can be observed, the learned MLD contains geometric patterns in the first few levels, stochastic textures in the last few levels and a combination of both in the middle levels. The representation energy, $\|\mathbf{\Psi}_l \mathbf{A}_l\|_F^2$, captured across all the levels in MLD is shown in Figure 5.3(a), where the energy hierarchy in learning can be clearly seen.

Given a multilevel dictionary, an $S-$sparse representation for a test sample can be evaluated using the M-OMP or the RM-OMP procedures described in Section 5.2.4. For the learned K-SVD and multilevel dictionaries, we computed the sparse codes for patches from a test dataset, by varying the desired sparsity. The test dataset consisted of $120,000$ non-overlapping 8×8 patches extracted from images in the BSDS test images. The illustration in Figure 5.3(b) shows the mean squared error (MSE) of



(a) (b)

Figure 5.4: Comparison of the MSE obtained with the BSDS test dataset using (a) MLD dictionaries trained at different noise levels (σ_{tr}), (b) K-SVD dictionaries trained at different noise levels (σ_{tr}).

the representation as a function of the number of non-zero coefficients. For the case of MLD, the results obtained using both the M-OMP and the RM-OMP schemes are shown. The OMP algorithm was employed to compute the sparse coefficients with the K-SVD dictionary. It can be observed that the MSE obtained using the M-OMP procedure is higher in all cases of sparsity, when compared to RM-OMP. Since the RM-OMP procedure considers dictionary atoms from the neighboring levels when computing a coefficient, it results in an improved generalization. When compared to K-SVD, multilevel dictionaries lead to a more accurate reconstruction when the sparsity level $S \geq 4$, which is the range typically used in several applications.

5.3 Learning Dictionaries with Noisy Training Data

In this section, we demonstrate that the proposed multilevel dictionary and the K-SVD dictionary exhibit similar behavior, under noisy training conditions.

The training data consisted of $50,000$ randomly chosen patches, of size 8×8, extracted from the grayscale images of the Berkeley segmentation dataset (BSDS) training dataset [1]. The mean value of each training patch was removed as a

preprocessing step. In order to analyze the performance of the MLD learning under noisy conditions, additive white Gaussian noise (AWGN) of standard deviation $\sigma_{tr} = \{0, 15, 25\}$ was added to the training data. The number of levels for learning the MLD was fixed at 16 and the error goal was fixed at $(1.15\sigma_{tr})^2 M$ [139], where M is the dimensionality of the training data. The reason for this is that with a probability of 0.75, the AWGN noise with a standard deviation σ_{tr} will lie within a range $[-1.15\sigma_{tr}, 1.15\sigma_{tr}]$. By setting the error goal as $(1.15\sigma_{tr})^2 M$, on an average we will have a very low chance of choosing noise as a part of the representation. For comparison, we designed K-SVD dictionaries of size 64×256 at different noise levels with the same error goal and a maximum sparsity of 16. For testing, AWGN of standard deviation $\sigma_{te} = \{0, 5, 10, 15, 20, 25\}$ was added to the test patches. Note that we did not perform any denoising here and only used the global dictionaries to generate representations for the noisy test data. For the case of MLD, we report only the results obtained using RM-OMP. The illustration in Figures 5.4(a) and 5.4(b) demonstrate that the performance degradation, with respect to noise level in the training data, for both K-SVD and MLD dictionaries are similar.

5.4 Stability and Generalization

In this section, the behavior of the proposed dictionary learning algorithm is considered from the viewpoint of algorithmic stability: the behavior of the algorithm with respect to the perturbations in the training set. It will be shown that the dictionary atoms learned by the algorithm from two different training sets whose samples are realized from the same probability space, become arbitrarily close to each other, as the number of training samples $T \to \infty$. Since the proposed MLD learning is equivalent to learning K-lines cluster centroids in multiple levels, the stability analysis of K-lines clustering [35] will be utilized in order to prove its stability. For each level of learning, the cases of single and multiple minimizers to the clustering objective will

be considered. Proving that the learning algorithm is stable will show that the global dictionaries learned from the data depend only on the probability space to which the training samples belong and not on the actual samples themselves, as $T \to \infty$. We also show that the MLD learning generalizes asymptotically, i.e., the difference between expected error and average empirical error in training approaches zero, as $T \to \infty$. Therefore, the expected error for novel test data, drawn from the same distribution as the training data, will be close to the average empirical training error.

The stability analysis of the MLD algorithm will be performed by considering two different dictionaries $\mathbf{\Psi}$ and $\mathbf{\Lambda}$ with L levels each. Each level consists of K_l dictionary atoms and the sub-dictionaries in each level are indicated by $\mathbf{\Psi}_l$ and $\mathbf{\Lambda}_l$ respectively. Note that the sub-dictionaries $\mathbf{\Psi}_l$ and $\mathbf{\Lambda}_l$ are the cluster centers learned using K-lines clustering on the training data for level l. The steps involved in proving the overall stability of the algorithm are: (a) showing that each level of the algorithm is stable in terms of $L_1(P)$ distance between the distortion functions, defined in (5.12), as the number of training samples $T \to \infty$ (Section 5.4.2), (b) proving that stability in terms of $L_1(P)$ distances indicates closeness of the centers of the two clusterings (Section 5.4.3), in terms of the metric defined in (5.14), and (c) showing that level-wise stability leads to overall stability of the dictionary learning algorithm (Section 5.4.4).

5.4.1 Stability Analysis of K-lines Clustering

Analyzing the stability of unsupervised clustering algorithms can be valuable in terms of understanding their behavior with respect to perturbations in the training set. These algorithms extract the underlying structure in the training data and the quality of clustering is determined by an accompanying cost function. As a result, any clustering algorithm can be posed as a Empirical Risk Minimization (ERM) procedure, by defining a hypothesis class of loss functions to evaluate the possible cluster

configurations and to measure their quality [107]. For example, K-lines clustering can be posed as an ERM problem over the distortion function class

$$\mathcal{G}_K = \left\{ g_{\mathbf{\Psi}}(\mathbf{y}) = d(\mathbf{y}, \boldsymbol{\psi}_j), j = \underset{l \in \{1,\cdots,K\}}{\operatorname{argmax}} |\mathbf{y}^T \boldsymbol{\psi}_l| \right\}. \tag{5.8}$$

The class $\mathcal{G}_K$ is obtained by taking functions $g_{\mathbf{\Psi}}$ corresponding to all possible combinations of K unit length vectors from the $\mathbb{R}^M$ space for the set $\mathbf{\Psi}$. Let us define the probability space for the data in $\mathbb{R}^M$ as $(\mathcal{Y}, \mathbf{\Sigma}, P)$, where $\mathcal{Y}$ is the sample space and $\mathbf{\Sigma}$ is a sigma-algebra on $\mathcal{Y}$, i.e., the collection of subsets of $\mathcal{Y}$ over which the probability measure P is defined. The training samples, $\{\mathbf{y}_i\}_{i=1}^T$, are T i.i.d. realizations from the probability space.

Ideally, we are interested in computing the cluster centroids $\hat{\mathbf{\Psi}}$ that minimize the expected distortion $\mathbb{E}[g_{\mathbf{\Psi}}]$ with respect to the probability measure P. However, the underlying distribution of the data samples is not known and hence we resort to minimizing the average empirical distortion with respect to the training samples $\{\mathbf{y}_i\}_{i=1}^T$ as

$$g_{\hat{\mathbf{\Psi}}} = \underset{g \in \mathcal{G}_K}{\operatorname{argmin}} \frac{1}{T} \sum_{i=1}^{T} g_{\mathbf{\Psi}}(\mathbf{y}_i). \tag{5.9}$$

When the empirical averages of the distortion functions in $\mathcal{G}_K$ uniformly converge to the expected values over all probability measures P,

$$\lim_{T \to \infty} \sup_P \mathbb{P}\left(\sup_{g_{\mathbf{\Psi}} \in \mathcal{G}_K} \left| \mathbb{E}[g_{\mathbf{\Psi}}] - \frac{1}{T} \sum_{i=1}^{T} g_{\mathbf{\Psi}}(\mathbf{y}_i) \right| > \delta \right) = 0, \tag{5.10}$$

for any $\delta > 0$, we refer to the class $\mathcal{G}_K$ as uniform Glivenko-Cantelli (uGC). In addition to being uGC, if the class also satisfies a version of the central limit theorem, it is defined as uniform Donsker [108]. In order to determine if $\mathcal{G}_K$ is uniform Donsker, we have to verify if the covering number of $\mathcal{G}_K$ with respect to the supremum norm, $N_\infty(\gamma, \mathcal{G}_K)$, grows polynomially in the dimensions M [26]. Here, γ denotes the maximum L_∞ distance between an arbitrary distortion function in $\mathcal{G}_K$, and the function that covers it. For K-lines clustering, the covering number is upper bounded

by [35, Lemma 2.1]

$$N_\infty(\gamma, \mathcal{G}_K) \leq \left(\frac{8R^3K + \gamma}{\gamma}\right)^{MK}, \tag{5.11}$$

where we assume that the data lies in an M-dimensional ℓ_2 ball of radius R centered at the origin. Therefore, $\mathcal{G}_K$ belongs to the uniform Donsker class.

The general idea behind stability of a clustering algorithm is that the algorithm should produce cluster centroids that are not significantly different when different i.i.d. training sets from the same probability space are used for training [24–26]. Stability is characterized based on the number of minimizers to the clustering objective with respect to the underlying data distribution. A minimizer corresponds to a function $g_{\mathbf{\Psi}} \in \mathcal{G}_K$ with the minimum expectation $\mathbb{E}[g_{\mathbf{\Psi}}]$. Stability analysis of the K-means algorithm has been reported in [25, 26].

Though the geometry of K-lines clustering is different from that of K-means, the stability characteristics of the two clustering algorithms have been found to be similar [35]. Given two sets of cluster centroids $\mathbf{\Psi} = \{\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_K\}$ and $\mathbf{\Lambda} = \{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_K\}$ learned from training sets of T i.i.d. samples each realized from the same probability space, let us define the $L_1(P)$ distance between the corresponding clusterings as

$$\|g_{\mathbf{\Psi}} - g_{\mathbf{\Lambda}}\|_{L_1(P)} = \int |g_{\mathbf{\Psi}}(\mathbf{y}) - g_{\mathbf{\Lambda}}(\mathbf{y})| dP(\mathbf{y}). \tag{5.12}$$

When $T \to \infty$, and $\mathcal{G}_K$ is uniform Donsker, stability in terms of the distortion functions is expressed as

$$\|g_{\mathbf{\Psi}} - g_{\mathbf{\Lambda}}\|_{L_1(P)} \xrightarrow{P} 0, \tag{5.13}$$

where $\xrightarrow{P}$ denotes convergence in probability. This holds true even for $\mathbf{\Psi}$ and $\mathbf{\Lambda}$ learned from completely disjoint training sets, when there is a unique minimizer to the clustering objective. When there are multiple minimizers, (5.13) holds true with respect to a change in $o(\sqrt{T})$ samples between two training sets and fails to hold with respect to a change in $\Omega(\sqrt{T})$ samples [35]. The distance between the cluster

centroids themselves is defined as [26]

$$\Delta(\mathbf{\Psi}, \mathbf{\Lambda}) = \max_{1 \le j \le K} \min_{1 \le l \le K} \left[(d(\boldsymbol{\psi}_j, \boldsymbol{\lambda}_l))^{1/2} + (d(\boldsymbol{\psi}_l, \boldsymbol{\lambda}_j))^{1/2} \right]. \tag{5.14}$$

Lemma 5.4.1 ([35]) *If the $L_1(P)$ distance between the distortion functions for the clusterings $\mathbf{\Psi}$ and $\mathbf{\Lambda}$ is bounded as $\|g_{\mathbf{\Psi}} - g_{\mathbf{\Lambda}}\|_{L_1(P)} < \mu$, for some $\mu > 0$, and $dP(\mathbf{y})/d\mathbf{y} > C$, for some $C > 0$, then $\Delta(\mathbf{\Psi}, \mathbf{\Lambda}) \le 2\sin(\rho)$ where*

$$\rho \le 2 \sin^{-1} \left[\frac{1}{r} \left(\frac{\mu}{\hat{C}_{C,M}} \right)^{\frac{1}{M+1}} \right]. \tag{5.15}$$

Here the training data is assumed to lie outside an M-dimensional ℓ_2 ball of radius r centered at the origin, and the constant $\hat{C}_{C,M}$ depends only on C and M.

When the clustering algorithm is stable according to (5.13), for admissible values of r, Lemma 5.4.1 indicates that the cluster centroids become arbitrarily close to each other, $\Delta(\mathbf{\Psi}, \mathbf{\Lambda}) \xrightarrow{P} 0$, which implies stability in terms of cluster centroids. From (5.15), it is also clear that the K-lines clustering cannot be stable if some training vectors have a norm close enough to 0, (i.e.) $r \to 0$.

5.4.2 Level-wise Stability for MLD Learning

Let us define a probability space $(\mathcal{Y}_l, \mathbf{\Sigma}_l, P_l)$ where $\mathcal{Y}_l$ is the data that lies in $\mathbb{R}^M$, and P_l is the probability measure. The training samples for the sub-dictionaries $\mathbf{\Psi}_l$ and $\mathbf{\Lambda}_l$ are two different sets of T i.i.d. realizations from the probability space. We also assume that the ℓ_2 norm of the training samples is bounded from above and below (i.e.), $0 < r \le \|\mathbf{y}\|_2 \le R < \infty$. Note that, in a general case, the data will lie in $\mathbb{R}^M$ for the first level of dictionary learning and in a finite union of lower-dimensional subspaces of $\mathbb{R}^M$ for the subsequent levels. In both cases, the following argument on stability will hold. This is because when the training data lies in a union of lower dimensional subspaces of $\mathbb{R}^M$, we can assume it to be still lying in $\mathbb{R}^M$, but assign the probabilities outside the union of subspaces to be zero.

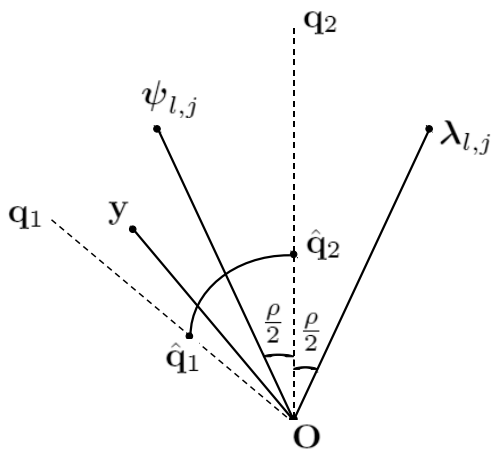


Figure 5.5: Illustration for showing the stability of cluster centroids from the stability of distortion function.

In each level, $\mathbf{\Psi}_l$ and $\mathbf{\Lambda}_l$ are learned using the K-lines clustering algorithm on two different i.i.d. sets of training data. The distortion function class for the clusterings, defined similar to (5.8), is uniform Donsker because the covering number with respect to the supremum norm grows polynomially, according to (5.11). When a unique minimizer exists for the clustering objective, the distortion functions corresponding to the different clusterings $\mathbf{\Psi}_l$ and $\mathbf{\Lambda}_l$ become arbitrarily close, $\|g_{\mathbf{\Psi}_l} - g_{\mathbf{\Lambda}_l}\|_{L_1(P_l)} \xrightarrow{P} 0$, even for completely disjoint training sets, as $T \to \infty$. However, in the case of multiple minimizers, $\|g_{\mathbf{\Psi}_l} - g_{\mathbf{\Lambda}_l}\|_{L_1(P_l)} \xrightarrow{P} 0$ holds only with respect to a change of $o(\sqrt{T})$ training samples between the two clusterings, and fails to hold when there is a change of $\Omega(\sqrt{T})$ samples [26, 35].

5.4.3 Distance between Cluster Centers for a Stable Clustering

For each cluster center in the clustering $\mathbf{\Psi}_l$, we pick the closest cluster center from $\mathbf{\Lambda}_l$, in terms of the distortion measure (3.1), and form pairs. Let us indicate the j^{th} pair of cluster centers by $\boldsymbol{\psi}_{l,j}$ and $\boldsymbol{\lambda}_{l,j}$. Let us define τ disjoint sets $\{A_i\}_{i=1}^{\tau}$, in which the training data for the clusterings exist, such that $P_l(\cup_{i=1}^{\tau} A_i) = 1$. By defining such disjoint sets, we can formalize the notion of training data lying in a union of subspaces of $\mathbb{R}^M$. The intuitive fact that the cluster centers of two clusterings are

close to each other in $\mathbb{R}^M$ space, given that their distortion functions are close, is proved in the lemma below.

Lemma 5.4.2 *Consider two sub-dictionaries (clusterings) $\mathbf{\Psi}_l$ and $\mathbf{\Lambda}_l$ with K_l atoms each obtained using the T training samples that exist in the τ disjoint sets $\{A_i\}_{i=1}^{\tau}$ in the $\mathbb{R}^M$ space, with $0 < r \leq \|\mathbf{y}\|_2 \leq R < \infty$, and $dP_l(\mathbf{y})/d\mathbf{y} > C$ in each of the sets. When the distortion functions become arbitrarily close to each other, $\|g_{\mathbf{\Psi}_l} - g_{\mathbf{\Lambda}_l}\|_{L_1(P_l)} \xrightarrow{P} 0$ as $T \to \infty$, the smallest angle between the subspaces spanned by the cluster centers becomes arbitrarily close to zero, i.e.,*

$$\angle(\boldsymbol{\psi}_{l,j}, \boldsymbol{\lambda}_{l,j}) \xrightarrow{P} 0, , \forall j \in 1, \ldots, K_l. \tag{5.16}$$

Proof Denote the smallest angle between the subspaces represented by $\boldsymbol{\psi}_{l,j}$ and $\mathbf{\Lambda}_{l,j}$ as $\angle(\boldsymbol{\psi}_{l,j}, \mathbf{\Lambda}_{l,j}) = \rho_{l,j}$ and define a region $S(\boldsymbol{\psi}_{l,j}, \rho_{l,j}/2) = \{\mathbf{y} | \angle(\boldsymbol{\psi}_{l,j}, \mathbf{y}) \leq \rho_{l,j}/2, 0 < r \leq \|\mathbf{y}\|_2 \leq R < \infty\}$. If $\mathbf{y} \in S(\boldsymbol{\psi}_{l,j}, \rho_{l,j}/2)$, then $\mathbf{y}^T(\mathbf{I} - \boldsymbol{\psi}_{l,j}\boldsymbol{\psi}_{l,j}^T)\mathbf{y} \leq \mathbf{y}^T(\mathbf{I} - \boldsymbol{\lambda}_{l,j}\boldsymbol{\lambda}_{l,j}^T)\mathbf{y}$. An illustration of this setup for a 2-D case is given in Figure 5.5. In this figure, the arc $\hat{\mathbf{q}}_1\hat{\mathbf{q}}_2$ is of radius r and represents the minimum value of $\|\mathbf{y}\|_2$. By definition, the $L_1(P_l)$ distance between the distortion functions of the clusterings for data that exists in the disjoint sets $\{A_i\}_{i=1}^{\tau}$ is

$$\|g_{\mathbf{\Psi}_l} - g_{\mathbf{\Lambda}_l}\|_{L_1(P_l)} = \sum_{i=1}^{\tau} \int_{A_i} |g_{\mathbf{\Psi}_l}(\mathbf{y}) - g_{\mathbf{\Lambda}_l}(\mathbf{y})| dP_l(\mathbf{y}). \tag{5.17}$$

For any j and i with a non-empty $B_{l,i,j} = S(\boldsymbol{\psi}_{l,j}, \rho_{l,j}/2) \cap A_i$ we have,

$$\|g_{\mathbf{\Psi}_l} - g_{\mathbf{\Lambda}_l}\|_{L_1(P_l)} \geq \int_{B_{l,i,j}} |g_{\mathbf{\Psi}_l}(\mathbf{y}) - g_{\mathbf{\Lambda}_l}(\mathbf{y})| dP_l(\mathbf{y}), \tag{5.18}$$

$$= \int_{B_{l,i,j}} [\mathbf{y}^T \left(\mathbf{I} - \boldsymbol{\lambda}_{l,j}\boldsymbol{\lambda}_{l,j}^T\right)\mathbf{y} - \sum_{k=1}^{K} \mathbf{y}^T \left(\mathbf{I} - \boldsymbol{\psi}_{l,k}\boldsymbol{\psi}_{l,k}^T\right)\mathbf{y} \mathbb{I}\left(\mathbf{y} \text{ closest to } \boldsymbol{\psi}_{l,k}\right)] dP_l(\mathbf{y}), \tag{5.19}$$

$$\geq \int_{B_{l,i,j}} [\mathbf{y}^T \left(\mathbf{I} - \boldsymbol{\lambda}_{l,j}\boldsymbol{\lambda}_{l,j}^T\right)\mathbf{y} - \mathbf{y}^T \left(\mathbf{I} - \boldsymbol{\psi}_{l,j}\boldsymbol{\psi}_{l,j}^T\right)\mathbf{y}] dP_l(\mathbf{y}), \tag{5.20}$$

$$\geq C \int_{B_{l,i,j}} \left[\left(\mathbf{y}^T\boldsymbol{\psi}_{l,j}\right)^2 - \left(\mathbf{y}^T\boldsymbol{\lambda}_{l,j}\right)^2\right] d\mathbf{y}. \tag{5.21}$$

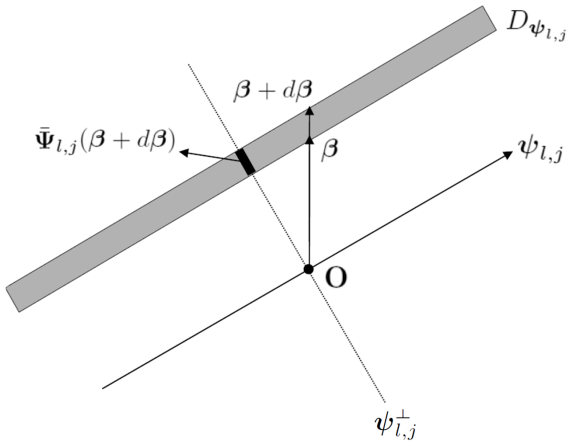


Figure 5.6: The residual set $\{\bar{\boldsymbol{\Psi}}_{l,j}(\boldsymbol{\beta}+d\boldsymbol{\beta})\}$, for the 1-D subspace $\boldsymbol{\psi}_{l,j}$, lying in its orthogonal complement subspace $\boldsymbol{\psi}_{l,j}^{\perp}$.

We have $g_{\boldsymbol{\Lambda}_l}(\mathbf{y}) = \mathbf{y}^T\left(\mathbf{I}-\boldsymbol{\lambda}_{l,j}\boldsymbol{\lambda}_{l,j}^T\right)\mathbf{y}$ in (5.19), since $\boldsymbol{\lambda}_{l,j}$ is the closest cluster center to the data in $S(\boldsymbol{\psi}_{l,j},\rho_{l,j}/2)\cap A_i$ in terms of the distortion measure (3.1). Note that $\mathbb{I}$ is the indicator function and (5.21) follows from (5.20) because $dP_l(\mathbf{y})/d\mathbf{y} > C$. Since by assumption, $\|g_{\boldsymbol{\Psi}_l} - g_{\boldsymbol{\Lambda}_l}\|_{L_1(P_l)} \xrightarrow{P} 0$, from (5.21), we have

$$\left(\mathbf{y}^T\boldsymbol{\psi}_{l,j}\right)^2 - \left(\mathbf{y}^T\boldsymbol{\lambda}_{l,j}\right)^2 \xrightarrow{P} 0, \tag{5.22}$$

because the integrand in (5.21) is a continuous non-negative function in the region of integration.

Denoting the smallest angles between $\mathbf{y}$ and the subspaces spanned by $\boldsymbol{\psi}_{l,j}$ and $\boldsymbol{\lambda}_{l,j}$ to be $\theta_{\boldsymbol{\psi}_{l,j}}$ and $\theta_{\boldsymbol{\lambda}_{l,j}}$ respectively, from (5.22), we have $\|\mathbf{y}\|_2^2(\cos^2\theta_{\boldsymbol{\psi}_{l,j}} - \cos^2\theta_{\boldsymbol{\lambda}_{l,j}}) \xrightarrow{P} 0$, for all $\mathbf{y}$. By definition of the region $B_{l,i,j}$, we have $\theta_{\boldsymbol{\psi}_{l,j}} \le \theta_{\boldsymbol{\lambda}_{l,j}}$. Since $\|\mathbf{y}\|_2$ is bounded away from zero and infinity, if $(\cos^2\theta_{\boldsymbol{\psi}_{l,j}} - \cos^2\theta_{\boldsymbol{\lambda}_{l,j}}) \xrightarrow{P} 0$ holds for all $\mathbf{y}\in B_{l,i,j}$, then we have $\angle(\boldsymbol{\psi}_{l,j},\boldsymbol{\lambda}_{l,j}) \xrightarrow{P} 0$. This is true for all cluster center pairs as we have shown this for an arbitrary i and j.

5.4.4 Stability of the MLD Algorithm

The stability of the MLD algorithm as a whole, is proved in Theorem 5.4.4 from its levelwise stability by using an induction argument. The proof will depend on the

following lemma which shows that the residuals from two stable clusterings belong to the same probability space.

Lemma 5.4.3 *When the training vectors for the sub-dictionaries (clusterings)* $\mathbf{\Psi}_l$ *and* $\mathbf{\Lambda}_l$ *are obtained from the probability space* $(\mathcal{Y}_l, \mathbf{\Sigma}_l, P_l)$*, and the cluster center pairs become arbitrarily close to each other as* $T \to \infty$*, the residual vectors from both the clusterings belong to an identical probability space* $(\mathcal{Y}_{l+1}, \mathbf{\Sigma}_{l+1}, P_{l+1})$*.*

Proof For the j^{th} cluster center pair $\boldsymbol{\psi}_{l,j}$, $\boldsymbol{\lambda}_{l,j}$, define $\bar{\mathbf{\Psi}}_{l,j}$ and $\bar{\mathbf{\Lambda}}_{l,j}$ as the projection matrices for their respective orthogonal complement subspaces $\boldsymbol{\psi}_{l,j}^{\perp}$ and $\boldsymbol{\lambda}_{l,j}^{\perp}$. Define the sets $D_{\boldsymbol{\psi}_{l,j}} = \{\mathbf{y} \in \bar{\mathbf{\Psi}}_{l,j}(\boldsymbol{\beta} + d\boldsymbol{\beta}) + \boldsymbol{\psi}_{l,j}\alpha\}$ and $D_{\boldsymbol{\lambda}_{l,j}} = \{\mathbf{y} \in \bar{\mathbf{\Lambda}}_{l,j}(\boldsymbol{\beta} + d\boldsymbol{\beta}) + \boldsymbol{\lambda}_{l,j}\alpha\}$, where $-\infty < \alpha < \infty$, $\boldsymbol{\beta}$ is an arbitrary fixed vector, not orthogonal to both $\boldsymbol{\psi}_{l,j}$ and $\boldsymbol{\lambda}_{l,j}$, and $d\boldsymbol{\beta}$ is a differential element. The residual vector set for the cluster $\boldsymbol{\psi}_{l,j}$, when $\mathbf{y} \in D_{\boldsymbol{\psi}_{l,j}}$ is given by, $\mathbf{r}_{\boldsymbol{\psi}_{l,j}} \in \{\bar{\mathbf{\Psi}}_{l,j}\mathbf{y} | \mathbf{y} \in D_{\boldsymbol{\psi}_{l,j}}\}$, or equivalently $\mathbf{r}_{\boldsymbol{\psi}_{l,j}} \in \{\bar{\mathbf{\Psi}}_{l,j}(\boldsymbol{\beta} + d\boldsymbol{\beta})\}$. Similarly for the cluster $\boldsymbol{\lambda}_{l,j}$, we have $\mathbf{r}_{\boldsymbol{\lambda}_{l,j}} \in \{\bar{\mathbf{\Lambda}}_{l,j}(\boldsymbol{\beta} + d\boldsymbol{\beta})\}$. For a 2-D case, Figure 5.6 shows the 1-D subspace $\boldsymbol{\psi}_{l,j}$, its orthogonal complement $\boldsymbol{\psi}_{l,j}^{\perp}$, the set $D_{\boldsymbol{\psi}_{l,j}}$ and the residual set $\{\bar{\mathbf{\Psi}}_{l,j}(\boldsymbol{\beta} + d\boldsymbol{\beta})\}$.

In terms of probabilities, we also have that $P_l(\mathbf{y} \in D_{\boldsymbol{\psi}_{l,j}}) = P_{l+1}(\mathbf{r}_{\boldsymbol{\psi}_{l,j}} \in \{\bar{\mathbf{\Psi}}_{l,j}(\boldsymbol{\beta} + d\boldsymbol{\beta})\})$, because the residual set $\{\bar{\mathbf{\Psi}}_{l,j}(\boldsymbol{\beta} + d\boldsymbol{\beta})\}$ is obtained by a linear transformation of $D_{\boldsymbol{\psi}_{l,j}}$. Here P_l and P_{l+1} are probability measures defined on the training data for levels l and $l+1$ respectively. Similarly, $P_l(\mathbf{y} \in D_{\boldsymbol{\lambda}_{l,j}}) = P_{l+1}(\mathbf{r}_{\boldsymbol{\lambda}_{l,j}} \in \{\bar{\mathbf{\Lambda}}_{l,j}(\boldsymbol{\beta} + d\boldsymbol{\beta})\})$. When $T \to \infty$, the cluster center pairs become arbitrarily close to each other, i.e., $\angle(\boldsymbol{\psi}_{l,j}, \boldsymbol{\lambda}_{l,j}) \xrightarrow{P} 0$, by assumption. Therefore, the symmetric difference between the sets $D_{\boldsymbol{\psi}_{l,j}}$ and $D_{\boldsymbol{\lambda}_{l,j}}$ approaches the null set, which implies that $P_l(\mathbf{y} \in D_{\boldsymbol{\psi}_{l,j}}) - P_l(\mathbf{y} \in D_{\boldsymbol{\lambda}_{l,j}}) \to 0$. This implies,

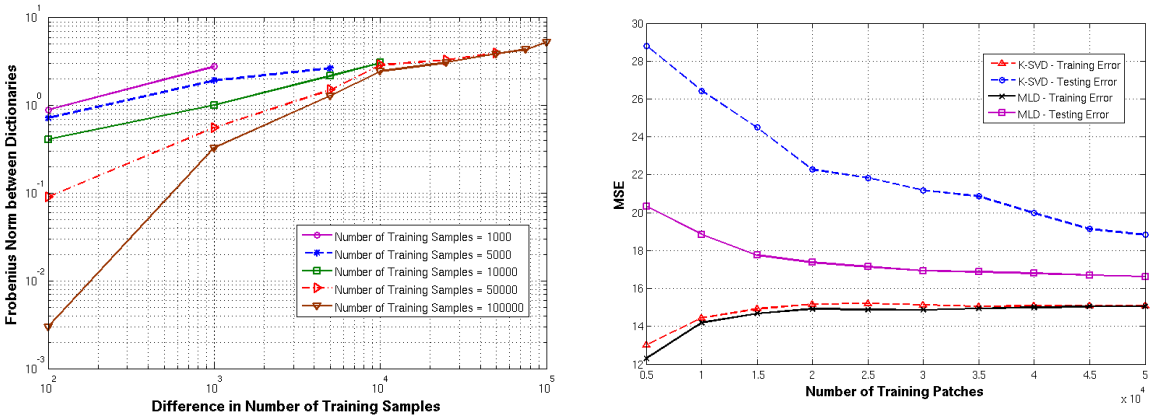
$$P_{l+1}(\mathbf{r}_{\boldsymbol{\psi}_{l,j}} \in \{\bar{\mathbf{\Psi}}_{l,j}(\boldsymbol{\beta} + d\boldsymbol{\beta})\}) - P_{l+1}(\mathbf{r}_{\boldsymbol{\lambda}_{l,j}} \in \{\bar{\mathbf{\Lambda}}_{l,j}(\boldsymbol{\beta} + d\boldsymbol{\beta})\}) \to 0, \tag{5.23}$$

for an arbitrary $\boldsymbol{\beta}$ and $d\boldsymbol{\beta}$, as $T \to \infty$. This means that the residuals of $\boldsymbol{\psi}_{l,j}$ and $\boldsymbol{\lambda}_{l,j}$ belong to a unique but identical probability space. Since we proved this for an arbitrary l and j, we can say that the residuals of clusterings $\boldsymbol{\Psi}_l$ and $\boldsymbol{\Lambda}_l$ belong to an identical probability space given by $(\mathcal{Y}_{l+1}, \boldsymbol{\Sigma}_{l+1}, P_{l+1})$.

Theorem 5.4.4 *Given that the training vectors for the first level are generated from the probability space* $(\mathcal{Y}_1, \boldsymbol{\Sigma}_1, P_1)$*, and the norms of training vectors for each level are bounded as* $0 < r \leq \|\mathbf{y}\|_2 \leq R < \infty$*, the MLD learning algorithm is stable as a whole.*

Proof The level-wise stability of MLD was shown in Section 5.4.2, for two cases: (a) when a unique minimizer exists for the distortion function and (b) when a unique minimizer does not exist. Lemma 5.4.2 proved that the stability in terms of closeness of distortion functions implied stability in terms of learned cluster centers. For showing the level-wise stability, we assumed that the training vectors in level l for clusterings $\boldsymbol{\Psi}_l$ and $\boldsymbol{\Lambda}_l$ belonged to the same probability space. However, when learning the dictionary, this is true only for the first level, as we supply the algorithm with training vectors from the probability space $(\mathcal{Y}_1, \boldsymbol{\Sigma}_1, P_1)$.

We note that the training vectors for level $l+1$ are residuals of the clusterings $\boldsymbol{\Psi}_l$ and $\boldsymbol{\Lambda}_l$. Lemma 5.4.3 showed that the residuals of level l for both the clusterings belong to an identical probability space $(\mathcal{Y}_{l+1}, \boldsymbol{\Sigma}_{l+1}, P_{l+1})$, given that the training vectors of level l are realizations from the probability space $(\mathcal{Y}_l, \boldsymbol{\Sigma}_l, P_l)$ and $T \to \infty$. By induction, this along with the fact that the training vectors for level 1 belong to the same probability space $(\mathcal{Y}_1, \boldsymbol{\Sigma}_1, P_1)$, shows that all the training vectors of both the dictionaries for any level l indeed belong to a probability space $(\mathcal{Y}_l, \boldsymbol{\Sigma}_l, P_l)$ corresponding to that level. Hence all the levels of the dictionary learning are stable and the MLD learning is stable as a whole.



(a) (b)

Figure 5.7: (a) Demonstration of the stability behavior of the proposed MLD learning algorithm. The minimum Frobenius norm between difference of two dictionaries with respect to permutation of their columns and signs is shown. The second dictionary is obtained by replacing different number of samples in the training set, used for training the original dictionary, with new data samples. (b) Demonstration of the generalization characteristics of the proposed algorithm compared to K-SVD. We plot the MSE obtained by representing patches from the BSDS test dataset, using dictionaries learned with different number of training patches. For comparison, we show the training error obtained in each case.

If there is a unique minimizer to the clustering objective in all levels of MLD learning, then the MLD algorithm is stable even for completely disjoint training sets, as $T \rightarrow \infty$. However, if there are multiple minimizers in at least one level, the algorithm is stable only with respect to a change of $o(\sqrt{T})$ training samples between the two clusterings. In particular, a change in $\Omega(\sqrt{T})$ samples makes the algorithm unstable.

5.4.5 Generalization Analysis

Since our learning algorithm consists of multiple levels, and cannot be expressed as an ERM on a whole, the algorithm can be said to generalize asymptotically if the sum of empirical errors for all levels converge to the sum of expected errors, as the

number of training samples $T \to \infty$. This can be expressed as

$$\left| \frac{1}{T} \sum_{l=1}^{L} \sum_{i=1}^{T} g_{\mathbf{\Psi}_l}(\mathbf{y}_{l,i}) - \sum_{l=1}^{L} \mathbb{E}_{P_l}[g_{\mathbf{\Psi}_l}] \right| \xrightarrow{P} 0, \tag{5.24}$$

where the training samples for level l given by $\{\mathbf{y}_{l,i}\}_{i=1}^{T}$ are obtained from the probability space $(\mathcal{Y}_l, \mathbf{\Sigma}_l, P_l)$. When (5.24) holds and the learning algorithm generalizes, it can be seen that the expected error for test data which is drawn from the same probability space as that of the training data, is close to the average empirical error. Therefore, when the cluster centers for each level are obtained by minimizing the empirical error, we are guaranteed that the expected test error will also be small.

In order to show that (5.24) holds, we use the fact that each level of MLD learning is obtained using K-lines clustering. Hence, from (5.10), the average empirical distortion in each level converges to the expected distortion as $T \to \infty$,

$$\left| \frac{1}{T} \sum_{i=1}^{T} g_{\mathbf{\Psi}_l}(\mathbf{y}_{l,i}) - \mathbb{E}_{P_l}[g_{\mathbf{\Psi}_l}] \right| \xrightarrow{P} 0. \tag{5.25}$$

The validity of the condition in (5.24) follows directly from the triangle inequality,

$$\begin{aligned} &\left| \frac{1}{T} \sum_{l=1}^{L} \sum_{i=1}^{T} g_{\mathbf{\Psi}_l}(\mathbf{y}_{l,i}) - \sum_{l=1}^{L} \mathbb{E}_{P_l}[g_{\mathbf{\Psi}_l}] \right| \\ &\leq \sum_{l=1}^{L} \left| \frac{1}{T} \sum_{i=1}^{T} g_{\mathbf{\Psi}_l}(\mathbf{y}_{l,i}) - \mathbb{E}_{P_l}[g_{\mathbf{\Psi}_l}] \right|. \end{aligned} \tag{5.26}$$

If the M-OMP coding scheme is used for test data, and the training and test data for level 1 are obtained from the probability space $(\mathcal{Y}_1, \mathbf{\Sigma}_1, P_1)$, the probability space for both training and test data in level l will be $(\mathcal{Y}_l, \mathbf{\Sigma}_l, P_l)$. This is because, both the M-OMP coding scheme and the MLD learning associate the data to a dictionary atom using the maximum absolute correlation measure and create a residual that is orthogonal to the atoms chosen so far. Hence, the assumption that training and test data are drawn from the same probability space in all levels hold and the expected test error will be similar to the average empirical training error.

5.4.6 Simulations

Both stability and generalization are crucial for building effective global dictionaries to model natural image patches. Although it is not possible to demonstrate the asymptotic behavior experimentally, we study the changes in the behavior of the learning algorithm with increase in the number of samples used for training.

In order to illustrate the stability characteristics of MLD learning, we setup an experiment where we consider a multilevel dictionary of 4 levels, with 8 atoms in each level. We extracted patches of size 8×8 from the BSDS training images and trained multilevel dictionaries using different number of training patches T. As we showed in Section 5.4, asymptotic stability is guaranteed when the training set is changed by not more than $o(\sqrt{T})$ samples. In other words, the inferred dictionary atoms will not vary significantly, if this condition is satisfied.

We fixed the size of the training set at different values $T = \{1000,\ 5000,\ 10000,\ 50000,\ 100000\}$ and learned an initial set of dictionaries using the proposed algorithm. The second set of dictionaries were obtained by replacing different number of samples from the original training set. For each case of T, the number of replaced samples was varied between 100 and T. For example, when $T = 10000$, the number of replaced training samples were $100, 1000, 5000$, and 10000. The amount of change between the initial and the second set of dictionaries was quantified using the minimum Frobenius norm of their difference with respect to permutations of their columns and sign changes. In Figure 5.7(a), we plot this quantity for different values of T as a function of the number of samples replaced in the training set. For each case of T, the difference between the dictionaries increases as we increase the replaced number of training samples. Furthermore, for a fixed number of replaced samples (say 100), the difference reduces with the increase in the number of training samples, since it becomes closer to asymptotic behavior.

In order to demonstrate the generalization characteristics of MLD learning, we designed dictionaries using different number of training image patches, of size 8×8, from the BSDS training dataset and evaluated the sparse codes for patches in the BSDS test dataset (Section 5.2.5). The dictionaries were learned at 16 levels with 16 atoms per level. Figure 5.7(b) shows the approximation error (MSE) for both the training and test datasets obtained using multilevel dictionaries. Furthermore, the corresponding MSE for the case of similarly designed K-SVD dictionaries are included for comparison. In all cases, the sparsity in training and testing was fixed at $S = 16$. As it can be observed, with MLD, the difference between the MSE for training and test data is small even for a small training set. However, the K-SVD dictionaries resulted in much higher MSE difference for a small training set, although the MSE with training data is similar for both MLD and KSVD. Note that, in both cases, the approximation error for the test data reduces with the increase in the size of the training set.

5.5 Robust Multilevel Dictionaries

The finite size of the training set and the lack of robustness in the initialization of K-lines clustering can affect the generalization of multilevel dictionaries to novel test patches. This is evidenced by the improved performance of RM-OMP over level-wise approximation (M-OMP). A common approach to achieving robustness in learning is to build an ensemble of multiple models inferred using different random subsets of the training data. *Bagging* (bootstrap aggregating) is a well known machine learning techniqdietterich2000experimentalue [146] that uses bootstrapping (equiprobable selection of samples with replacement) on the training data to create many varied training sets with overlapping samples. It then evaluates the hypothesis for each bootstrap sample, and creates an ensemble by averaging the different hypotheses. This has been successfully used in several classification and regression problems.

Table 5.2: PSNR (dB) of the images recovered from compressed measurements obtained using Gaussian random measurement matrices. Results obtained using the proposed MLD, RMLD-Ex and RMLD dictionaries, along with K-SVD, are shown for different measurement noise conditions and number of measurements. Higher PSNR for each case is indicated in bold font.

Measurement SNR (dB)	**Method**	**Image**									
		Barbara		**Boat**		**House**		**Lena**		**Man**	
		N=16	**N=32**	**N=16**	**N=32**	**N=16**	**N=32**	**N=16**	**N=32**	**N=16**	**N=32**
0	K-SVD (OMP)	20.54	21.51	22.07	23.42	23.91	25.48	24.23	26.16	23.2	24.9
	MLD (RM-OMP)	20.63	21.9	22.38	23.56	23.98	25.54	24.51	26.43	23.59	25.18
	RMLD-Ex (M-OMP)	21.99	22.56	23.76	24.19	24.93	26.4	25.35	26.03	25.14	25.61
	RMLD (M-OMP)	**22.02**	**22.6**	**23.76**	**24.25**	**24.97**	**26.44**	**25.38**	**26.11**	**25.15**	**25.66**
15	K-SVD (OMP)	21.89	24.34	25.02	27.39	26.87	31.01	28.08	31.42	26.02	28.54
	MLD (RM-OMP)	22.41	24.95	25.26	27.83	27.15	31.37	28.29	31.55	26.19	28.82
	RMLD-Ex (M-OMP)	24.09	26.12	26.37	29.17	28.32	31.13	28.94	31.27	27.47	29.75
	RMLD (M-OMP)	**24.16**	**26.17**	**26.69**	**29.48**	**28.79**	**31.38**	**29.11**	**31.4**	**27.62**	**30.04**
25	K-SVD (OMP)	22.09	24.99	25.88	28.7	27.1	31.6	29.03	31.83	26.59	29.36
	MLD (RM-OMP)	22.62	25.26	25.27	28.82	27.31	31.78	28.64	32	26.71	29.59
	R-MLD-Ex (M-OMP)	24.14	26.57	26.57	29.52	28.44	32.18	29.12	32.19	27.61	30
	R-MLD (M-OMP)	**24.33**	**26.72**	**27.07**	**29.68**	**29.04**	**32.38**	**29.55**	**32.36**	**27.79**	**30.38**

Table 5.3: Average Time(seconds) taken in MATLAB for training dictionaries, with $50,000$ samples, and recovering images of size 512×512 using different number of random measurements.

Method	**Training**	**N = 16**	**N = 32**
MLD	502	0.11	0.19
RMLD-Ex	90	2.6	5.6
RMLD	1980	1.05	2.31

We propose to employ bagging to reduce overfitting in multilevel dictionary learning and thereby improve the accuracy of the dictionaries in representing novel test samples. Though it is typical to aggregate categorical values or numerical outputs, we propose to aggregate the signal approximation in each level of an MLD. Instead of employing K-lines clustering on the training set (T samples), we draw D bootstrap samples, i.e., we construct D training subsets each containing T_D samples ($T_D << T$). For each subset, we learn K dictionary atoms using K-lines clustering. Note that, we allow overlap between the different bootstrap samples. For each training sample in that level, we compute $1-$sparse representations using all the D dictionaries. The signal approximation in level 1 of the MLD is computed as the average of approximations using all D dictionaries, $\frac{1}{D}\sum_d \mathbf{\Psi}_1^d \mathbf{A}_1^d$. Here, the superscript d denotes the dictionary and coefficient matrices corresponding to round d in level 1 of the MLD. The general idea of learning atoms at multiple levels, with varying levels of complexity, can be extended to this case as well. The ensemble representations are used to compute the set of residuals and this process is repeated for a desired number of levels. We refer to this dictionary as a Robust MLD (RMLD).

Since we employ bagging, the $1-$sparse approximation in each level is more robust and can generalize well to novel test samples. Hence, by adopting a simple level-wise approximation scheme, Robust MLD can perform better than the RM-

(a) K-SVD (26.25 dB)

(b) MLD (26.59 dB)

(c) BMLD-Ex (27.26 dB)

(d) BMLD (27.81 dB)

Figure 5.8: Compressed recovery of images from random measurements ($N = 16$, SNR of measurement process = 15dB) using the different dictionaries. In each case the PSNR of the recovered image is also shown.

OMP scheme with an MLD. The robustness in performance comes at the price of increased complexity for computing D $1-$sparse representations in each level (Table 5.3). Furthermore, the ensemble coefficient vector will not be $1-$sparse as in MLD and hence cannot be efficient for storage-constrained applications such as compression. However, the significant improvement in approximation achieved by bagged multilevel dictionaries merits their use in several image recovery problems. Though it is beyond the scope of this work, we note that more efficient schemes to construct the multiple dictionaries from the training set can be designed, and stability of the resulting ensemble sparse codes can be analyzed.

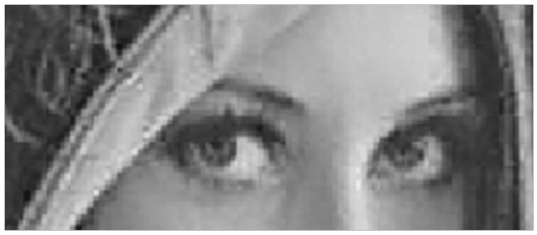
Table 5.4: PSNR (dB) of the images recovered from compressed measurements obtained using optimized measurement matrices. The performance of the proposed MLD dictionary is compared with that of K-SVD for different measurement noise conditions and number of measurements. Higher PSNR for each case is indicated in bold font.

Measurement SNR (dB)	**Method**	**Image**									
		Barbara		**Boat**		**House**		**Lena**		**Man**	
		N=16	**N=32**	**N=16**	**N=32**	**N=16**	**N=32**	**N=16**	**N=32**	**N=16**	**N=32**
0	K-SVD (OMP)	21.52	22.28	22.25	24.48	23.33	25.89	24.04	26.67	22.87	25.2
	MLD (RM-OMP)	**22.1**	**22.61**	**23.47**	**24.75**	**24.82**	**26.31**	**25.55**	**27.03**	**24.48**	**25.56**
15	K-SVD (OMP)	24.39	28.18	27.93	31.71	30.2	35.11	30.45	35.12	28.48	31.95
	MLD (RM-OMP)	**25.17**	**28.35**	**29.54**	**32.63**	**32.07**	**35.38**	**32.59**	**35.61**	**30.33**	**33.36**
25	K-SVD (OMP)	24.79	29.18	28.54	33.01	31.1	35.98	31.32	36.2	29.17	33.6
	MLD (RM-OMP)	**25.43**	**29.2**	**30.38**	**34.56**	**33.3**	**38.67**	**33.89**	**38.29**	**31.16**	**35.23**

(a) K-SVD recovery (PSNR = 28.47 dB)

(b) MLD recovery (PSNR = 30.31 dB)

(c) K-SVD recovery (PSNR = 30.43 dB)



(d) MLD recovery (PSNR = 32.6 dB)

Figure 5.9: Compressed recovery of images using optimized measurements ($N = 16$, SNR of measurement process = 15dB). Only small portions of the images are displayed for visualizing the differences in the quality of recovery.

Furthermore, the complexity of performing K-lines clustering for D rounds in each level is quite high. This can be simplified by constructing simple dictionaries using the training samples directly. In each level, we randomly choose K samples from the training set as the dictionary atoms and normalize them to unit ℓ_2 norm. We refer to this dictionary as BMLD-Ex. For a fixed value of D, the complexity of approximating a test sample is the same for RMLD and RMLD-Ex dictionaries. More details about ensemble dictionary learning and its use in MLD design can be found in [147, 148].

5.6 Application: Compressed Recovery

In this section, we demonstrate the performance of multilevel dictionaries in compressed sensing of images using the BSDS dataset and a set of standard images, with random and optimized measurement systems. Sensing and recovery were performed on a patch-by-patch basis, on non-overlapping patches of size 8×8. MLD and K-SVD dictionaries were learned with 50000 BSDS patches as described in Section 5.2.5. For

learning the BMLD, we fix $K = 16$ and learn $D = 20$ rounds of K-lines dictionaries in each level ($L = 16$) using random sets of training data. We also observed the BMLD-Ex needed more rounds of dictionaries in each level when compared to BMLD and hence we fixed D at 50 in that case. The measurement process can be expressed as

$$\mathbf{x} = \mathbf{\Phi}\mathbf{\Psi}\mathbf{a} + \boldsymbol{\eta}, \tag{5.27}$$

where $\mathbf{\Psi}$ is the dictionary (MLD or K-SVD in our case), $\mathbf{\Phi}$ is the measurement or projection matrix, $\boldsymbol{\eta}$ is the AWGN vector added to the measurement process, $\mathbf{x}$ is the output of the measurement process and $\mathbf{a}$ is the sparse coefficient vector such that $\mathbf{y} = \mathbf{\Psi}\mathbf{a}$. The size of the data vector $\mathbf{y}$ is $M \times 1$, that of $\mathbf{\Psi}$ is $M \times K$, that of the measurement matrix $\mathbf{\Phi}$ is $N \times M$, where $N < M$, and that of the measured vector $\mathbf{x}$ is $N \times 1$. The entries in the random measurement matrices were independent realizations from a standard normal distribution. We recover the underlying image from its compressed measurements, using the K-SVD, MLD, and BMLD dictionaries. For each case, we present average results from 100 trial runs, each time with a different measurement matrix.

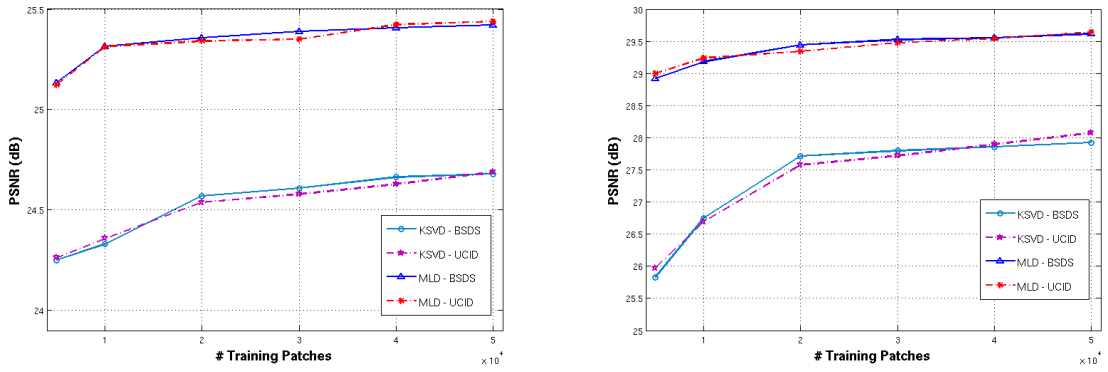
We evaluated the recovery performance for the following set of standard images: *Barbara*, *Boat*, *House*, *Lena*, and *Man*. Table 5.2 shows the PSNR (peak signal-to-noise ratio) obtained using different number of measurements, and at varying measurement noise levels. As it can be observed, the proposed multilevel dictionaries outperform the K-SVD dictionaries in all cases. Furthermore, the proposed Bagged MLD algorithm results in much improved recovery, for a slight increase in complexity. The average time taken for recovering an image of size 512×512 using the three proposed dictionaries are listed in Table 5.3. Figure 5.9 illustrates the recovered images obtained using different dictionaries with random measurements.

Besides performing sensing with random measurement matrices, that are used extensively in compressed sensing, we also used measurement matrices optimized to

dictionaries that can provide much better performance than random matrices [140, 149]. We adopt the strategy proposed in [140], and compute optimized measurement matrices for the pre-existing K-SVD and multilevel dictionaries. Note that, we did not compute optimized measurements for the Bagged MLD since it is not feasible to compute the measurements for the several dictionaries in each level. Similar to the previous case, multilevel dictionaries achieve superior recovery results in comparison to K-SVD dictionaries (Table 5.4).

5.6.1 Variation of Recovery Performance with the Training Set

In order to demonstrate the dependence of the proposed MLD learning on the size and type of the training set, we performed compressed recovery of the *Boat* image using dictionaries learned using different number of training patches obtained from two different standard image databases: the BSDS [1] and the UCID [150]. The number of training image patches, obtained from each of the datasets, was varied between $5,000$ and $50,000$. For each case, we learned an MLD with 16 levels and 16 atoms per level, and a K-SVD dictionary of size 64×256 with the number of non-zero coefficients (S) fixed at 16. We performed compressed recovery using 16 random and optimized measurements at a measurement SNR of 15 dB. For random measurements the results were averaged over 100 iterations. Figures 5.10(a) and 5.10(b) show the recovery performance obtained for each dataset using both the dictionaries, for random and optimized projections respectively. As it can be observed, the performance is fairly independent of the dataset used for training. Furthermore, the proposed learning scheme achieves a better performance, as well as a much smaller performance variations, across dictionaries obtained using different number of training samples.



(a) (b)

Figure 5.10: Compressed recovery performance of the *Boat* image with K-SVD and MLD dictionaries learned using different number of training patches randomly chosen from different standard datasets for, (a) random measurements and (b) optimized measurements.

5.7 Application: Denoising

Our goal in denoising is to recover the clean image Y from the noisy observed image X. The image X is divided into patches of size 8×8 with an overlap of 1 pixel, and these patches are vectorized and stacked in the matrix $\mathbf{X}$. A noisy observation $\mathbf{x}$ (a column in $\mathbf{X}$), can be represented as a corrupted version of its corresponding clean patch, $\mathbf{x} = \mathbf{y} + \boldsymbol{\eta}$, where $\boldsymbol{\eta}$ is the AWGN vector with standard deviation σ. Patchwise recovery was performed using RM-OMP with the global MLD dictionary learned with $50,000$ patches as described in Section 5.2.5. Patchwise error goal was fixed and image-level reconstruction constraints were posed as described in [139]. All results were averaged over 5 iterations. Note that, under low-noise conditions dictionaries learned from the noisy test image itself perform better than global dictionaries. However, under high-noise conditions, global dictionaries perform comparably to image-specific dictionaries. This results in a significant computational advantage since it is not necessary to train a separate dictionary for each noisy image. Furthermore, we focus on global dictionary learning in this chapter and hence we compare the results of

Table 5.5: PSNR (dB) of the denoised standard images corrupted with AWGN of standard deviation σ. In each case, the average of 5 trials is provided. Higher performance is shown in bold font.

Noise	**Image**									
	Barbara		**Boat**		**Fingerprint**		**House**		**Lena**	
	K-SVD	**MLD**	**K-SVD**	**MLD**	**K-SVD**	**MLD**	**K-SVD**	**MLD**	**K-SVD**	**MLD**
20	28.87	**29.15**	30.24	**30.31**	28.21	**28.37**	32.88	**32.93**	32.27	**32.44**
25	27.57	**27.91**	29.17	**29.26**	26.94	**27.22**	31.82	**31.99**	31.2	**31.37**
50	24.06	**24.15**	25.91	**25.97**	22.68	**23.36**	27.91	**27.98**	27.77	**27.89**
75	22.54	**22.57**	24.02	**24.06**	19.73	**20.24**	25.33	**25.42**	25.81	**25.92**
100	**21.73**	21.72	22.83	**22.92**	18.23	**18.72**	23.86	**24.06**	24.45	**24.51**

Table 5.6: Average Time(seconds) taken in MATLAB for denoising images of size 512×512 under different noise conditions.

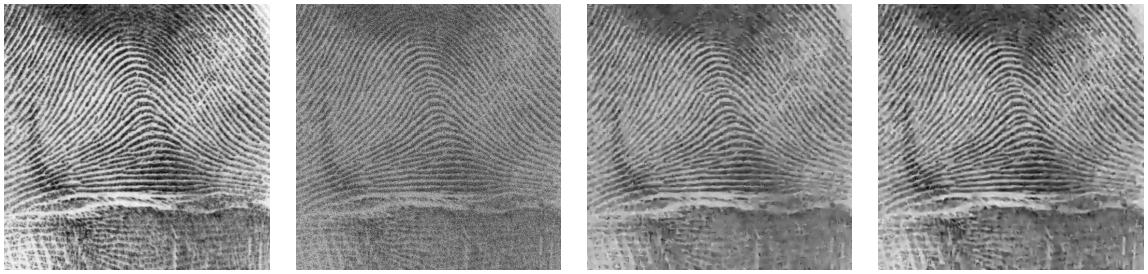
Method	$\sigma = 20$	$\sigma = 25$	$\sigma = 50$	$\sigma = 75$	$\sigma = 100$
K-SVD [4]	16.47	14.93	11.43	9.58	8.61
MLD	8.79	8.52	7.96	7.54	7.11

global MLD and K-SVD dictionaries in Table 5.5 for high-noise conditions ($\sigma \geq 20$). For global K-SVD dictionary, the results reported in [139] were used. It can be seen that, in almost all the cases, global MLD performs better than global K-SVD dictionaries. The denoised *Lena* and *Fingerprint* images are shown in Figure 5.11, for $\sigma = 20$ and $sigma = 50$ respectively, where a clear improvement in reconstruction performance is observed. Computationally, denoising using MLD is less expensive compared to using K-SVD as seen from Table 5.6. All the times reported in this chapter are obtained using MATLAB 2012a on a 2.8 GHz, 8-core Intel i7 Linux machine.

5.8 Application: Image Compression

In this section, we report the performance of multilevel dictionaries in image compression. MLD and K-SVD dictionaries of size 64×512 learned under zero training noise conditions were employed for compression of a test image. The test image illus-

(a) Original (b) Noisy (c) K-SVD (d) MLD



(a) Original (b) Noisy (c) K-SVD (d) MLD

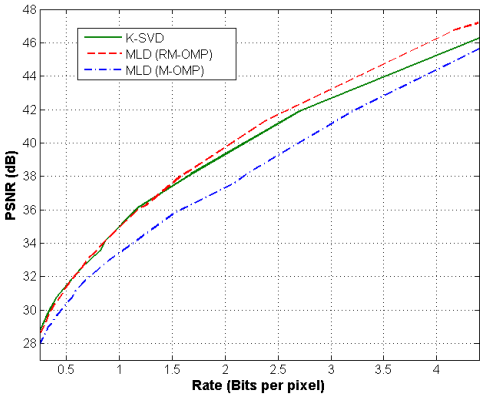
Figure 5.11: Original, noisy and denoised *Lena* and *Fingerprint* images with their respective PSNRs. Reconstructed images for global K-SVD dictionaries are obtained using the *K-SVD toolbox* [4].

trated in Figure 5.12(a) was obtained from the Uncompressed Color Image Database (UCID) [150] and converted to grayscale. The multilevel dictionary was obtained from the BSDS training set ($50,000$ patches) with 16 atoms per level and 32 levels. A K-SVD dictionary of size 64×512 was learned from the training set, with the maximum sparsity fixed at 32. The simulation setup used is similar to the one presented in [83]. The rate of compression in bits per pixel is given by,

$$R = \frac{a_1 B + F(a_2 + Q)}{P}, \tag{5.28}$$

where a_1 is the number of bits required to code the number of coefficients per patch, B is the total number of patches coded, F is the total number of coefficients to be coded for the full test dataset, a_2 is the number of bits required to code the index of each coefficient, Q is the number of bits required to code a single coefficient and P is the total number of pixels in the test set. Assuming entropy coding, the values

(a)



(b)

Figure 5.12: (a) Test image from the UCID dataset for compression, (b) Rate-Distortion curves obtained using the MLD and K-SVD global dictionaries of 512 atoms.

of a_1 and a_2 are $\log_2 32$ and $\log_2 512$ as there could be a maximum of 32 coefficients per patch and the total number of indices for the coefficients are 512. In this case, we removed the mean value from each patch and then coded the patches. However, coding the mean values will not affect our performance comparison, as the extra rate required to code the mean will only shift the RD curves to the right.

In order to obtain different rates, the maximum possible sparsity was fixed at 32 and the error goals for the pursuit algorithms were varied. The coefficients obtained for each error goal were then quantized using uniform quantization for different values of Q. For quantization purposes, a single upper and lower bound for the coefficients was obtained using the training set. The MSE and hence the PSNR of reconstruction using the quantized coefficients were then collected. From the scatter plot of rate versus PSNR, the best values were then chosen and plotted as the RD curve. The curves thus obtained are given in Figure 5.12(b). As expected, performing M-OMP based reconstruction using MLD provides the worst results at all bit-rates. At low bit-rates, the performance of the K-SVD and the MLD dictionaries (RM-OMP) are

quite similar. However, at moderate to high bit rates MLD performs consistently better. Hence, it is clear that the MLD is well suited for compression at all bit rates.

Chapter 6

MULTIPLE KERNEL SPARSE REPRESENTATIONS

6.1 Problem Statement

Sparse models have been effective in several image recovery applications, and this has motivated their use in computer vision problems. One of the first sparse coding based object recognition frameworks used codes obtained from raw image patches [98]. However, since then several frameworks have been proposed that use sparse codes of local descriptors, such as the Scale Invariant Feature Transform (SIFT), extracted from images. In order to construct image level descriptors, the codes are aggregated in an orderless bag-of-features approach [151] or using the spatial pyramid matching (SPM) approach [99] that partially preserves the spatial ordering. Methods that use sparse codes of local descriptors in spatial pyramids have achieved better performance [100, 152–154], in comparison to the original SPM approach which is based on vector quantization. Furthermore, the authors in [100] demonstrated that spatial pyramid aggregation of sparse codes can lead to high object recognition accuracies with just linear classifiers.

Though sparsity can lead to improved separability in high-dimensional spaces [29], it has been experimentally shown that incorporating sparse coding for feature extraction does not guarantee an improvement in object recognition performance [155]. Hence, there is a need to promote discrimination implicitly or explicitly while performing sparse coding and dictionary learning for recognition tasks. For this reason, some algorithms explicitly incorporate class-specific discriminatory information when learning the dictionary, and this is applied for digit recognition and image classification [91, 152, 156, 157]. Furthermore, improved discrimination among classes have been obtained by performing simultaneous sparse coding to enforce similar non-zero coefficient support for all samples in a class [46, 158, 159], and coding the descriptors using dictionary atoms in their neighborhood [153]. Other approaches that lead to

discriminative sparse codes are those that constrain the sparse codes to obey the constraints induced by the neighborhood graphs of the training data [160, 161].

In addition to their widespread applicability in supervised learning frameworks, sparsity has also been shown to be useful in unsupervised clustering applications. In [162], the authors show that clustering graph-regularized sparse codes with K-means results in better clustering performance when compared to using the data directly. The relation between data samples can be inferred by representing each data sample as a sparse linear combination of all the others. These sparse codes can be then used to build an ℓ_1 graph, and spectral clustering can be performed on the graph [34]. However, if there are a large number of training examples, computing sparse codes in this manner can be quite expensive in terms of computations. In [163], the authors show that this can be avoided by using a fixed size dictionary with appropriate constraints. This dictionary can be then used to obtain sparse codes and they can be subsequently used to perform spectral clustering.

Despite its great applicability, the use of sparse models in image classification presents two main challenges. Firstly, in recognition application, no single descriptor can efficiently represent the various aspects of the data. Hence, there is a need to integrate multiple descriptors extracted from images into the sparse coding paradigm. The other challenge is that each descriptor could potentially belong to a different space and the similarity measure for each descriptor could be defined as a non-linear function. For example, descriptors can be expressed as vectors, matrices, scalar values, or tensors. The proper way to combine them hence would be to fuse the information that each descriptor provides about the data and not the raw descriptors themselves. This can be efficiently performed by using appropriate kernel functions, which measures the possible non-linear similarity between each set of descriptors [33], and combining the kernel similarities to perform sparse coding in the unified feature space. The non-linear similarities between descriptors in the input space transforms

to linear similarities in the unified feature space. The advantage of using multiple diverse features in object recognition has been demonstrated in a number of research efforts [164].

The sparse models learned in the unified feature space will lead to discriminatory codes that can be used with linear classifiers. This is because, in this space, the similarity measure between the features is linear and hence those that belong to the same class will be grouped in the same low-dimensional subspaces. Furthermore, since it is appropriate to use linear similarities linear classifiers are sufficient. Greedy approaches to obtain sparse codes using the kernel trick have been proposed [165], [166]. In [167], Guo *et.al.* proposed to perform kernel sparse coding of image descriptors, and design dictionaries for object recognition, when the Radial Basis Function (RBF) kernel is used. Furthermore, the authors of [168] designed a kernel dictionary learning algorithm for digit recognition and demonstrated improved discrimination, particularly in presence of noise.

6.2 Kernel Sparse Representations and Dictionary Learning

The kernel function maps the non-linearly separable features into a high dimensional feature space, in which similar features are grouped together, hence improving the linear separability [33]. The authors of [167] showed that sparse representations can be efficiently performed in a high dimensional feature space using kernel methods. In this section, we review the problem of kernel sparse representations, and describe the procedure to optimize dictionaries using a fixed point iteration method when the RBF kernel is used.

Let us define a function $\Phi : \mathbb{R}^M \mapsto \mathcal{F}$, that maps the data samples from the input space to a high dimensional feature space. The data sample in the input space $\mathbf{y}$ transforms to $\Phi(\mathbf{y})$ in the feature space and the N training examples given by $\mathbf{Y} = [\mathbf{y}_1 \dots \mathbf{y}_N]$ transform to $\Phi(\mathbf{Y}) = [\Phi(\mathbf{y}_1) \dots \Phi(\mathbf{y}_N)]$. The feature space similarity

or the kernel similarity between the training examples $\mathbf{y}_i$ and $\mathbf{y}_j$ are defined using the pre-defined kernel function as $\mathcal{K}(\mathbf{y}_i, \mathbf{y}_j) := \Phi(\mathbf{y}_i)^T \Phi(\mathbf{y}_j)$. The dictionary in the feature space is denoted by the matrix by $\Phi(\boldsymbol{\Psi}) = [\Phi(\boldsymbol{\psi}_1), \Phi(\boldsymbol{\psi}_2), ..., \Phi(\boldsymbol{\psi}_K)]$, where each column indicates a dictionary element. The similarities between dictionary elements and the training examples can also be computed using the kernel function as $\Phi(\boldsymbol{\psi}_k)^T \Phi(\mathbf{y}) = \mathcal{K}(\boldsymbol{\psi}_k, \mathbf{y}_k)$ and $\Phi(\boldsymbol{\psi}_k)^T \Phi(\boldsymbol{\psi}_l) = \mathcal{K}(\boldsymbol{\psi}_k, \boldsymbol{\psi}_l)$. Since all similarities can be computed exclusively using the kernel function, it is not necessary to know the transformation Φ. This greatly simplifies the computations in the feature space when the similarities are pre-computed and is referred to as the *kernel trick*. We use the notation $\mathbf{K}_{\mathbf{YY}} \in \mathbb{R}^{N \times N}$ to represent the matrix $\Phi(\mathbf{Y})^T \Phi(\mathbf{Y})$ and it contains the kernel similarities between all training examples. The similarity between two training examples given by $\mathcal{K}(\mathbf{y}_i, \mathbf{y}_j)$ is the $(i,j)^{\text{th}}$ element of $\mathbf{K}_{\mathbf{YY}}$, also denoted as $\mathbf{K}_{\mathbf{y}_i \mathbf{y}_j}$.

The problem of sparse coding in the feature space can be posed as the penalized ℓ_1 minimization

$$\min_{\mathbf{a}} \|\Phi(\mathbf{y}) - \Phi(\boldsymbol{\Psi})\mathbf{a}\|_2^2 + \lambda \|\mathbf{a}\|_1, \tag{6.1}$$

and the objective can be expanded as

$$\begin{aligned} &\Phi(\mathbf{y})^T \Phi(\mathbf{y}) - 2\mathbf{a}^T \Phi(\boldsymbol{\Psi})^T \Phi(\mathbf{y}) + \mathbf{a}^T \Phi(\boldsymbol{\Psi})^T \Phi(\boldsymbol{\Psi})\mathbf{a} + \lambda \|\mathbf{a}\|_1, \\ &= \mathbf{K}_{\mathbf{yy}} - 2\mathbf{a}^T \mathbf{K}_{\boldsymbol{\Psi}\mathbf{y}} + \mathbf{a}^T \mathbf{K}_{\boldsymbol{\Psi}\boldsymbol{\Psi}} \mathbf{a} + \lambda \|\mathbf{a}\|_1. \end{aligned} \tag{6.2}$$

Note that we have used the kernel trick here to simplify the computations. Following our notation, $\mathbf{K}_{\mathbf{yy}}$ is the element $\mathcal{K}(\mathbf{y}, \mathbf{y})$, $\mathbf{K}_{\boldsymbol{\Psi}\mathbf{y}}$ is a $K \times 1$ vector containing the elements $\mathcal{K}(\boldsymbol{\psi}_k, \mathbf{y})$, for $k = \{1, \ldots, K\}$ and $\mathbf{K}_{\boldsymbol{\Psi}\boldsymbol{\Psi}}$ is a $K \times K$ matrix containing the kernel similarities between all the dictionary atoms. As it can be easily observed, the modified objective function is similar to the sparse coding problem, except for the use of the kernel similarities. Hence, as noted in [167], the Feature-Sign Search algorithm can be employed to solve for the sparse codes. However, it is important to note that the computation of kernel matrices incurs additional complexity. Since

the dictionary is fixed in (7.3), $\mathbf{K}_{\boldsymbol{\Psi}\boldsymbol{\Psi}}$ is computed only once and the complexity of computing $\mathbf{K}_{\boldsymbol{\Psi}\mathbf{y}}$ grows as $O(MK)$. When the kernel sparse codes for all N training samples are computed, the dictionary can be obtained by minimizing

$$\sum_{i=1}^{N} \|\Phi(\mathbf{y}_i) - \Phi(\boldsymbol{\Psi})\mathbf{a}_i\|_2^2, \tag{6.3}$$

with respect to the constraint that they are normalized in the feature space, $\Phi(\boldsymbol{\psi}_k)^T\Phi(\boldsymbol{\psi}_k) = 1,\ \forall k = \{1, \ldots, K\}$. Since dictionary update is performed in the feature space, it is not straightforward to use standard procedures such as the K-SVD or Lagrangian dual based methods [64], and hence the authors in [167] employed a fixed point algorithm. In this section, we derive expressions for dictionary update using fixed point iteration method for the case of the RBF kernel. Note that, when the kernel function does not have a closed form or the function is not differentiable, this procedure cannot be used to update the dictionary. In Section 6.5, we describe a method that is suitable for dictionary learning with any arbitrary kernel matrix.

6.2.1 Radial Basis Function Kernel

The radial basis function (RBF) is a well-known kernel that has wide applicability and it is defined as

$$\mathcal{K}(\mathbf{y}_i, \mathbf{y}_j) = \exp(-\gamma\|\mathbf{y}_i - \mathbf{y}_j\|_2^2), \tag{6.4}$$

where γ is a positive constant. Note that, $\mathcal{K}(\boldsymbol{\psi}_k, \boldsymbol{\psi}_k) = 1$ and hence there is no need to enforce the normalization constraint $\Phi(\boldsymbol{\psi}_k)^T\Phi(\boldsymbol{\psi}_k) = 1$ when updating the dictionary atoms. The objective function for dictionary update in (6.3) can be simplified as

$$\sum_{i=1}^{N}\left[1 - 2\sum_{l=1}^{K} a_{l,i}\mathcal{K}(\boldsymbol{\psi}_l, \mathbf{y}_i) + \sum_{l=1}^{K}\sum_{t=1}^{K} a_{l,i}a_{t,i}\mathcal{K}(\boldsymbol{\psi}_l, \boldsymbol{\psi}_t)\right], \tag{6.5}$$

where $a_{l,i}$ represents the i^{th} element in the coefficient vector $\mathbf{a}_l$. In order to update the dictionary atom $\boldsymbol{\psi}_k$, we compute the derivative of (6.5) using the definition of RBF kernel given in (6.4), and set the derivative to zero as follows,

$$-4\gamma\sum_{i=1}^{N}\left[-a_{k,i}\mathcal{K}(\boldsymbol{\psi}_k, \mathbf{y}_i)(\boldsymbol{\psi}_k - \mathbf{y}_i) + \sum_{t=1}^{K} a_{k,i}a_{t,i}\mathcal{K}(\boldsymbol{\psi}_k, \boldsymbol{\psi}_t)(\boldsymbol{\psi}_k - \boldsymbol{\psi}_t)\right] = 0. \tag{6.6}$$

Since no closed-form expression can be obtained for updating the dictionary atom $\boldsymbol{\psi}_k$, we use the fixed point algorithm similar to the one proposed in [167]. In this procedure, the dictionary atom $\boldsymbol{\psi}_k$ from the $(n-1)^{\text{th}}$ is used to compute the kernel similarities for the n^{th} of the update. Denoting the k^{th} atom in the n^{th} iteration by $\boldsymbol{\psi}_k^{(n)}$, we can rewrite the expression in (6.6) as

$$-4\gamma\sum_{i=1}^{N}\left[-a_{k,i}\mathcal{K}(\boldsymbol{\psi}_k^{(n-1)},\mathbf{y}_i)(\boldsymbol{\psi}_k^{(n)}-\mathbf{y}_i)+\sum_{t=1}^{K}a_{k,i}a_{t,i}\mathcal{K}(\boldsymbol{\psi}_k^{(n-1)},\boldsymbol{\psi}_t)(\boldsymbol{\psi}_k^{(n)}-\boldsymbol{\psi}_t)\right]=0. \tag{6.7}$$

By solving (6.7), we obtain

$$\boldsymbol{\psi}_k^{(n)}=\frac{\boldsymbol{\Psi}diag[\mathbf{K}_{\boldsymbol{\psi}_k\boldsymbol{\Psi}}^{(n-1)}]\mathbf{A}\mathbf{a}_{k,row}^T-\mathbf{Y}diag[\mathbf{K}_{\boldsymbol{\psi}_k\mathbf{Y}}^{(n-1)}]\mathbf{a}_{k,row}^T}{\mathbf{K}_{\boldsymbol{\psi}_k\boldsymbol{\Psi}}^{(n-1)}\mathbf{A}\mathbf{a}_{k,row}^T-\mathbf{K}_{\boldsymbol{\psi}_k\mathbf{Y}}^{(n-1)}\mathbf{a}_{k,row}^T}, \tag{6.8}$$

Here $\mathbf{a}_{k,row}$ is a row vector containing the set of coefficients of all training vectors corresponding to the dictionary atom $\boldsymbol{\psi}_k$, and $diag[.]$ creates a diagonal matrix using the argument vector as its diagonal. The $1\times K$ kernel matrix $\mathbf{K}_{\boldsymbol{\psi}_k\boldsymbol{\Psi}}^{(n-1)}$ contains the elements $\mathcal{K}(\boldsymbol{\psi}_k^{(n-1)},\boldsymbol{\psi}_l)$, for $l=\{1\dots K\}$ and the $1\times N$ matrix $\mathbf{K}_{\boldsymbol{\psi}_k\mathbf{Y}}^{(n-1)}$ contains the elements $\mathcal{K}(\boldsymbol{\psi}_k^{(n-1)},\mathbf{y}_i)$, for $i=\{1\dots N\}$ respectively.

6.3 Multiple Kernel Sparse Representations

The use of multiple descriptors to characterize images has been a very successful approach for complex visual recognition tasks. Though this method provides the flexibility of choosing features to describe different aspects of the underlying data, the resulting representations are high-dimensional and the descriptors can be very diverse. Hence, in order to facilitate recognition tasks, there is a need to transform these features to a unified space, and construct low dimensional compact representations for the images in the unified space. Multiple Kernel Learning (MKL) provides a convenient way of fusing multiple descriptors by combining multiple base kernels, each of which is created based on an image descriptor [164]. In this work we develop the Multiple Kernel Sparse Representations (MKSR) model that aims to compute

the sparse representation for an image in the unified feature space obtained using multiple kernels. In addition to building a compact representation for the data in the high-dimensional feature space, MKSR can provide highly discriminative codes.

Suppose there are R descriptors extracted from each data sample. Let us denote the set of descriptors obtained from the data sample as $\mathbf{y}_i = \{\mathbf{y}_{i,r}\}_{r=1}^{R}$, where i is the sample index, and r is the descriptor index. Let the set of R base kernel functions, each corresponding to a descriptor be denoted as $\{\mathcal{K}_r\}_{r=1}^{R}$, and the base kernel matrices be denoted as $\{\mathbf{K}_r\}_{r=1}^{R}$. The ensemble kernel function and the ensemble kernel matrix can be constructed as the non-negative linear combination,

$$\mathcal{K}(\mathbf{y}_i, \mathbf{y}_j) = \sum_{r=1}^{R} \beta_r \mathcal{K}_r(\mathbf{y}_{i,r}, \mathbf{y}_{j,r}), \quad \forall \beta_r \geq 0, \tag{6.9}$$

$$\mathbf{K} = \sum_{r=1}^{R} \beta_r \mathbf{K}_r, \quad \forall \beta_r \geq 0. \tag{6.10}$$

As described earlier, the ensemble kernel matrix contains the similarities between the data samples in the unified feature space obtained using multiple kernels. Various types of descriptors such as raw pixel values, histograms, feature vectors, and spatial pyramids can be successfully combined by considering only the kernel matrices corresponding to the descriptors. In order to obtain the kernel matrix for a descriptor, we can either employ a pre-defined kernel function such as the RBF or create a user-defined kernel matrix using the pairwise distances between the descriptors. Given a distance function ρ_r, that measures the distance between the descriptors obtained from two samples, we can construct the kernel matrix as

$$\mathbf{K}_r(i,j) = \mathcal{K}_r(\mathbf{y}_{i,r}, \mathbf{y}_{j,r}) = \exp(-\gamma \rho_r^2(\mathbf{y}_{i,r}, \mathbf{y}_{j,r})), \tag{6.11}$$

where γ is a positive constant. Computing this kernel function is convenient, since several useful image descriptors and their corresponding distance functions have been proposed in the literature. Note that the kernel matrix in (7.9) is not guaranteed to be positive semidefinite, which is a required characteristic for a proper kernel matrix.

In this case, we can follow the approach in [169] where we compute the smallest eigenvalue of $\mathbf{K}_r$ and if it is negative, we add its absolute value to the diagonal of $\mathbf{K}_r$.

Given a dictionary $\mathbf{\Psi}$ and a kernel function $\mathcal{K}$, sparse codes can be obtained as described in Section 6.2. However in the MKSR model we have R different descriptors for each image. In order to obtain sparse codes using an ensemble kernel matrix, we need to optimize the dictionaries for the unified feature space. In this chapter, we consider two different approaches for obtaining multiple kernel sparse codes and designing dictionaries using the kernel trick, that will be described in Sections 6.4 and 6.5 respectively. The first approach uses alternating optimization to compute separate dictionaries for each descriptor and obtains sparse codes in the unified feature space using ensemble kernel matrices. The second approach starts from the ensemble kernel matrix for data and directly builds a multi-level dictionary along with a multi-level representation in the unified feature space. This approach does not require knowledge of the mathematical form of the individual kernels. In contrast, the first approach needs a mathematical form of the kernel for each descriptor. The advantage with the first approach is that optimized dictionaries for each descriptor, if previously available, can be readily used to initialize the learning algorithm and hence better performances can be obtained.

6.4 Proposed Method 1

In this section, we present the *Method 1*, that alternatively optimizes separate dictionaries for each image descriptor and obtains MKSRs by fusing the individual kernels. Consider a dataset of N data samples and R different descriptors to characterize them. The kernel function $\mathcal{K}_r$ for descriptor r can be pre-defined kernel or constructed using the distance function d_r. For the r^{th} descriptor, we use its corresponding samples, $\{\mathbf{y}_{i,r}\}_{i=1}^{N}$, to learn the dictionary $\mathbf{\Psi}_r = [\boldsymbol{\psi}_{1,r}, \boldsymbol{\psi}_{2,r}, \ldots, \boldsymbol{\psi}_{K,r}]$ that can sparsely represent the descriptor set in its feature space. Given the descriptors and the dictionaries,

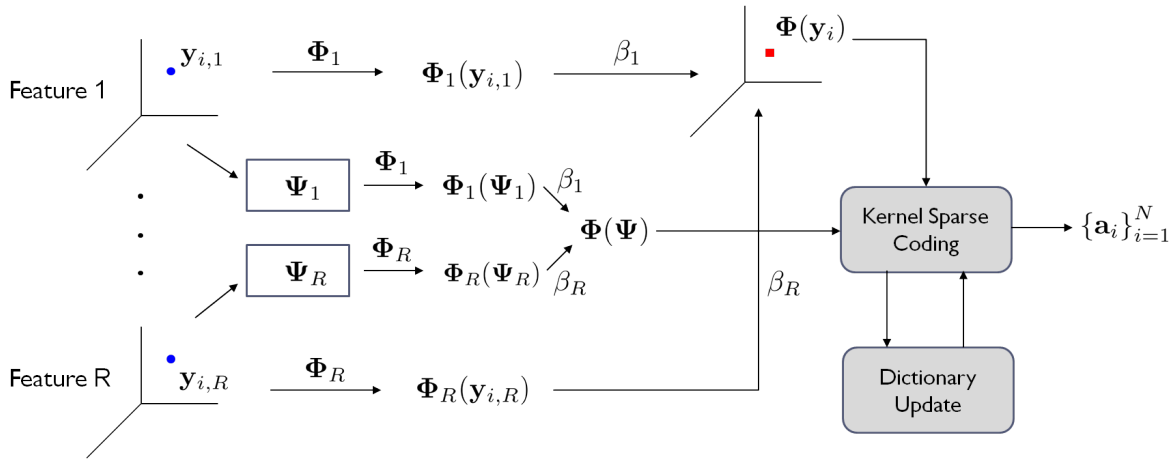


Figure 6.1: Proposed *Method 1* for obtaining multiple kernel sparse representations. In this approach, we alternatively optimize the individual dictionaries $\{\mathbf{\Psi}_r\}_{r=1}^R$ and obtain the unified sparse codes $\{\mathbf{a}_i\}_{i=1}^N$. Note that $r = \{1, \ldots, R\}$ denotes the index of the descriptor. The ensemble kernel matrices for both the data samples and the dictionary atoms are obtained as given in (7.10).

we compute the individual kernel matrices,

$$\mathbf{K}_{\mathbf{YY},r}(i,j) = \mathcal{K}_r(\mathbf{y}_{i,r}, \mathbf{y}_{j,r}) \quad \forall r = 1, \ldots, R, \tag{6.12}$$

$$\mathbf{K}_{\mathbf{\Psi Y},r}(k,i) = \mathcal{K}_r(\boldsymbol{\psi}_{k,r}, \mathbf{y}_{i,r}) \quad \forall r = 1, \ldots, R, \tag{6.13}$$

$$\mathbf{K}_{\mathbf{\Psi\Psi},r}(k,l) = \mathcal{K}_r(\boldsymbol{\psi}_{k,r}, \boldsymbol{\psi}_{l,r}) \quad \forall r = 1, \ldots, R. \tag{6.14}$$

Let us define the corresponding ensemble kernel matrices as

$$\mathbf{K}_{\mathbf{YY}}(i,j) = \sum_{r=1}^{R} \beta_r \mathbf{K}_{\mathbf{YY},r}(i,j), \forall \beta_r \geq 0. \tag{6.15}$$

$$\mathbf{K}_{\mathbf{\Psi Y}}(k,i) = \sum_{r=1}^{R} \beta_r \mathbf{K}_{\mathbf{\Psi Y},r}(i,j), \forall \beta_r \geq 0. \tag{6.16}$$

$$\mathbf{K}_{\mathbf{\Psi\Psi}}(k,l) = \sum_{r=1}^{R} \beta_r \mathbf{K}_{\mathbf{\Psi\Psi},r}(i,j), \forall \beta_r \geq 0. \tag{6.17}$$

Similar to the kernel sparse representations problem in (7.3), the objective function to be minimized for MKSR can be expressed as

$$\begin{aligned} &\min_{\mathbf{a}} \|\Phi(\mathbf{y}) - \Phi(\mathbf{\Psi})\mathbf{a}\|_2^2 + \lambda\|\mathbf{a}\|_1, \\ &= \mathbf{K}_{\mathbf{yy}} - 2\mathbf{a}^T\mathbf{K}_{\mathbf{\Psi y}} + \mathbf{a}^T\mathbf{K}_{\mathbf{\Psi\Psi}}\mathbf{a} + \lambda\|\mathbf{a}\|_1. \end{aligned} \tag{6.18}$$

Here $\Phi(\mathbf{y})$ and $\Phi(\mathbf{\Psi})$ denote the data sample and the dictionary in the multiple kernel domain. Given the ensemble kernel matrices, multiple kernel sparse codes can be efficiently obtained using the Feature-Sign search algorithm. The dictionaries and the MKSRs are hence computed in an alternating fashion until the objective is minimized.

6.4.1 Updating the Dictionaries

As illustrated in Figure 6.1, the proposed *Method 1* fuses the multiple dictionaries, using kernel matrices, to perform sparse coding in the unified feature space. Optimization of the dictionaries for efficient coding needs to be carried out using a fixed point algorithm as described in Section 6.2. Rewriting the objective function in (6.3) for dictionary update,

$$\sum_{i=1}^{N} \|\Phi(\mathbf{y}_i) - \Phi(\mathbf{\Psi})\mathbf{a}_i\|_2^2 \tag{6.19}$$

$$= \sum_{i=1}^{N} \left[\mathcal{K}(\mathbf{y}_i, \mathbf{y}_i) - 2\sum_{l=1}^{K} a_{l,i}\mathcal{K}(\boldsymbol{\psi}_l, \mathbf{y}_i) + \sum_{l=1}^{K}\sum_{t=1}^{K} a_{l,i}a_{t,i}\mathcal{K}(\boldsymbol{\psi}_l, \boldsymbol{\psi}_t) \right], \tag{6.20}$$

$$= \sum_{i=1}^{N} \left[\sum_{r=1}^{R} \beta_r \mathcal{K}_r(\mathbf{y}_{i,r}, \mathbf{y}_{i,r}) - 2\sum_{l=1}^{K} a_{l,i} \sum_{r=1}^{R} \beta_r \mathcal{K}_r(\boldsymbol{\psi}_{l,r}, \mathbf{y}_{i,r}) + \sum_{l=1}^{K}\sum_{t=1}^{K} a_{l,i}a_{t,i} \sum_{r=1}^{R} \beta_r \mathcal{K}_r(\boldsymbol{\psi}_{l,r}, \boldsymbol{\psi}_{t,r}) \right]. \tag{6.21}$$

As it can be observed in (6.21), we have expanded the ensemble kernel function in terms of its individual base kernels. Since there are R different dictionaries with K atoms each, the fixed point algorithm needs to update the atoms of one dictionary at a time, fixing the other $R-1$ dictionaries. Hence, to update the k^{th} atom of the $\mathbf{\Psi}_r$, we need to compute the derivative of (6.18) with respect to $\boldsymbol{\psi}_{k,r}$. Excluding all terms in (6.18) that do not depend on the current dictionary atom being updated, the objective can be simplified as

$$= \sum_{i=1}^{N} \left[-2\sum_{l=1}^{K} a_{l,i}\beta_r \mathcal{K}_r(\boldsymbol{\psi}_{l,r}, \mathbf{y}_{i,r}) + \sum_{l=1}^{K}\sum_{t=1}^{K} a_{l,i}a_{t,i}\beta_r \mathcal{K}_r(\boldsymbol{\psi}_{l,r}, \boldsymbol{\psi}_{t,r}) \right]. \tag{6.22}$$

The objective in (6.19) is equivalent to updating the dictionary atoms for kernel sparse coding, with the kernel function $\mathcal{K}_r$. Note that the same set of coefficients $\{\mathbf{a}_i\}_{i=1}^N$, are used for updating all the R dictionaries, since the representation is computed in the unified feature space. In order to obtain dictionary update expressions as shown in Section 6.2, the kernel function should be differentiable. Furthermore, the ensemble kernel weights $\{\beta_r\}_{r=1}^R$ balance the relative importance of each feature and hence they are chosen empirically such that best classification performance is obtained.

6.4.2 Computing Sparse Codes for Test Data

Given the test data sample $\mathbf{x}$, we extract the R different descriptors $\{\mathbf{x}_r\}_{r=1}^R$ and evaluate the ensemble kernel matrix $\mathbf{K}_{\mathbf{\Psi x}}$ of size $K \times 1$ where

$$\mathbf{K}_{\mathbf{\Psi x}}(k) = \sum_{r=1}^{R} \beta_r \mathcal{K}_r(\boldsymbol{\psi}_{k,r}, \mathbf{x}_r). \tag{6.23}$$

The kernel weights $\{\beta_r\}_{r=1}^R$ evaluated empirically during the training stage are used with the test data. The sparse code $\mathbf{a}_{\mathbf{x}}$ is then obtained by optimizing the objective

$$-2\mathbf{a}_{\mathbf{x}}^T \mathbf{K}_{\mathbf{\Psi x}} + \mathbf{a}_{\mathbf{x}}^T \mathbf{K}_{\mathbf{\Psi\Psi}} \mathbf{a}_{\mathbf{x}} + \lambda \|\mathbf{a}_{\mathbf{x}}\|_1. \tag{6.24}$$

6.5 Proposed Method 2

An alternative approach to optimizing the dictionary for sparse coding in the unified feature space is to perform dictionary learning directly using the ensemble kernel matrix of descriptors obtained from the training data. We learn a hierarchical dictionary in multiple levels and refer to this as kernel multilevel dictionary (MLD) learning. This algorithm is a generalization of the MLD algorithm proposed in [39] to kernel space. Such an approach eliminates the need to explicitly optimize separate dictionaries and hence provides the flexibility to choose any kernel function. The proposed framework, referred to as *Method 2*, is illustrated in Figure 6.2. In order to develop the kernel MLD algorithm, we first note that the problem of dictionary learning is a generalization of K-lines clustering [23], which fits K $1-$D subspaces to the set of

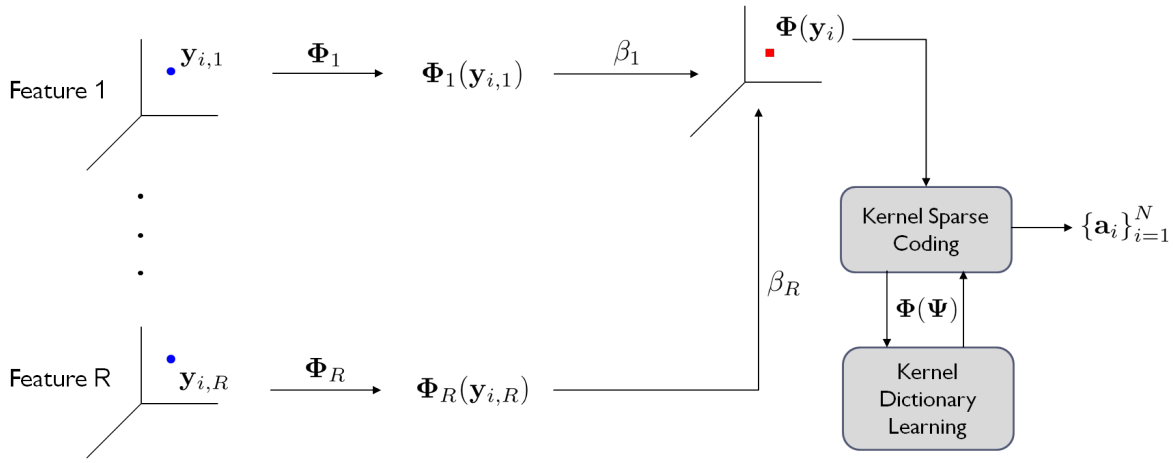


Figure 6.2: Proposed *Method 2* for obtaining multiple kernel sparse representations. In this approach, we evaluate the ensemble kernel matrix by fusing the base kernel matrices of the different image descriptors and perform dictionary learning in the unified feature space directly. The sparse code for a test sample is evaluated in the multiple kernel feature space using the learned dictionary.

data samples. In [39], it is shown that hierarchical dictionaries obtained by performing K-lines clustering in multiple levels are effective for sparse approximation. We begin by briefly discussing the K-lines clustering procedure and the multilevel dictionary (MLD) learning algorithm. We then present the kernelized version of the K-lines clustering algorithm [37] with some demonstrative results to show its superior clustering performance. Following this, we develop the kernel MLD learning algorithm for designing dictionaries using multiple kernels. Finally, we describe the procedure to compute the sparse code for a test sample using the kernel MLD.

6.5.1 Multilevel Dictionary Learning

The MLD learning algorithm proposed in [39] uses a hierarchical approach by employing K-lines clustering to adapt atoms in each level of the dictionary. We denote the multilevel dictionary as $\mathbf{\Psi} = [\mathbf{\Psi}_1 \mathbf{\Psi}_2 \dots \mathbf{\Psi}_L]$, and the coefficient matrix as $\mathbf{A} = [\mathbf{A}_1^T \mathbf{A}_2^T \dots \mathbf{A}_L^T]^T$. Here, $\mathbf{\Psi}_l$ is the sub-dictionary in level l and $\mathbf{A}_l$ is corresponding

the coefficient matrix in level l. The approximation in level l is expressed as

$$\mathbf{R}_{l-1} = \mathbf{\Psi}_l \mathbf{A}_l + \mathbf{R}_l, \text{ for } l = 1, ..., L, \tag{6.25}$$

where $\mathbf{R}_{l-1}$, $\mathbf{R}_l$ are the residuals for the levels $l-1$ and l respectively and $\mathbf{R}_0 = \mathbf{Y}$. This implies that the residuals in level $l-1$ serve as the training data for level l. Note that the sparsity of the representation in each level is fixed at 1. K-lines clustering is employed to learn $\mathbf{\Psi}_l$ from the training matrix, $\mathbf{R}_{l-1}$, for that level. Detailed discussion of the convergence and algorithmic stability of MLD learning along with its applications are reported in [40].

6.5.2 Kernel Multilevel Dictionary Learning Algorithm

Given a set of training samples, our goal is to design multilevel dictionaries in the unified feature space obtained using multiple kernels. The kernel K-lines clustering procedure developed in the previous section can be used to learn the atoms in every level of the dictionary. In level l, we denote the sub-dictionary by $\Phi(\mathbf{\Psi}_l)$, the membership matrix by $\mathbf{Z}_l$, the coefficient matrix by $\mathbf{A}_l$, the input and the residual matrices by $\Phi(\mathbf{Y}_l)$ and $\Phi(\mathbf{R}_l)$ respectively. Note that the training set for the first level $\mathbf{Y}_1 = \mathbf{Y}$. Given the N training images, we build the R descriptors and compute the ensemble kernel matrix $\mathbf{K_{YY}}$ as given in (6.15).

$$\mathbf{K_{YY}}(i,j) = \mathcal{K}(\mathbf{y}_i, \mathbf{y}_j) = \sum_{r=1}^{R} \beta_r \mathcal{K}_r(\mathbf{y}_{i,r} \mathbf{y}_{j,r}). \tag{6.26}$$

As described in the previous section, we can compute $\mathbf{H}_l = \mathbf{Z}_l \odot \mathbf{A}_l$. Performing kernel K-lines clustering in level 1 will yield the coefficients $\mathbf{A}_1 = \mathbf{K_{YY}} \mathbf{H}_1 \mathbf{D}_1$, where $\mathbf{D}_1 = \mathbf{\Gamma}(\Phi(\mathbf{Y}_1)\mathbf{H}_1)^{-1} = diag(\mathbf{H}_1^T \mathbf{K_{YY}} \mathbf{H}_1)^{-1/2}$ indicates the diagonal matrix that normalizes the dictionary atoms of level 1 in the feature space. In kernel MLD learning, the residual vectors in a level are used as the training set to the next level.

Hence, we compute the residuals of the training vectors as

$$\Phi(\mathbf{R}_1) = \Phi(\mathbf{Y}_1) - \Phi(\mathbf{\Psi}_1)\mathbf{H}_1^T \tag{6.27}$$

$$= \Phi(\mathbf{Y}_1) - \Phi(\mathbf{Y}_1)\mathbf{H}_1\mathbf{D}_1\mathbf{H}_1^T, \tag{6.28}$$

$$= \Phi(\mathbf{Y}_1)\left[\mathbf{I} - \mathbf{H}_1\mathbf{D}_1\mathbf{H}_1^T\right] = \Phi(\mathbf{Y}_2). \tag{6.29}$$

Note that we used the expression in (3.48) to obtain (6.28). Given the residuals from level 1, the dictionary atoms in level 2 can be computed as

$$\Phi(\mathbf{\Psi}_2) = \Phi(\mathbf{Y}_2)\mathbf{H}_2\mathbf{D}_2, \tag{6.30}$$

where

$$\mathbf{D}_2 = diag\left((\Phi(\mathbf{Y}_2)\mathbf{H}_2)^T(\Phi(\mathbf{Y}_2)\mathbf{H}_2)\right)^{-1/2}, \tag{6.31}$$

$$= diag[\mathbf{H}_2^T\left(\Phi(\mathbf{Y}_1)[\mathbf{I} - \mathbf{H}_1\mathbf{D}_1\mathbf{H}_1^T]\right)^T\left(\Phi(\mathbf{Y}_1)[\mathbf{I} - \mathbf{H}_1\mathbf{D}_1\mathbf{H}_1^T]\right)\mathbf{H}_2]^{-1/2}, \tag{6.32}$$

$$= diag[\mathbf{H}_2^T\left(\mathbf{I} - \mathbf{H}_1\mathbf{D}_1\mathbf{H}_1^T\right)^T\mathbf{K}_{\mathbf{YY}}\left(\mathbf{I} - \mathbf{H}_1\mathbf{D}_1\mathbf{H}_1^T\right)\mathbf{H}_2]^{-1/2}. \tag{6.33}$$

Similar to the previous level, the coefficients can be evaluated as

$$\mathbf{A}_2 = \Phi(\mathbf{Y}_2)^T\Phi(\mathbf{\Psi}_2), \tag{6.34}$$

$$= \left(\mathbf{I} - \mathbf{H}_1\mathbf{D}_1\mathbf{H}_1^T\right)^T\mathbf{K}_{\mathbf{YY}}\left(\mathbf{I} - \mathbf{H}_1\mathbf{D}_1\mathbf{H}_1^T\right)\mathbf{H}_2\mathbf{D}_2. \tag{6.35}$$

Table 6.1 shows the detailed algorithm to learn a kernel MLD by generalizing the procedure for S levels. Note that the innermost loop in the algorithm computes the cluster centroids using the linearized SVD procedure (Chapter 3). The middle loop performs the kernel K-lines clustering for a particular level. Similar to *Method 1*, the weights $\{\beta_r\}_{r=1}^R$ are tuned empirically to provide the best classification performance.

6.5.3 Computing Sparse Codes for Test Data

In this section, we describe a procedure to evaluate the sparse code for a test sample using the kernel MLD. Using the R descriptors $\{\mathbf{x}_r\}_{r=1}^R$ extracted from test sample,

Table 6.1: Kernel Multilevel Dictionary Learning algorithm.

Input
$\mathbf{Y}_1 = [\mathbf{y}_{1,i}]_{i=1}^N$, $D \times N$ matrix of training image patches.
$\mathbf{K_{YY}}$, $N \times N$ kernel matrix.
K, desired number of atoms per level.
S, total number of levels.

Initialization
- Randomly initialize the membership $\mathbf{Z}_1$ and compute the initial coefficients, $\mathbf{A}_1$.

Algorithm
For $s = 1$ **to** S
  Loop until convergence
    - Loop for L iterations
      - Compute $\mathbf{H}_s = \mathbf{Z}_s \odot \mathbf{A}_s$.
      - Compute $\mathbf{D}_s = diag\left[\mathbf{H}_s^T \left(\prod_{t=1}^{s-1}(\mathbf{I} - \mathbf{H}_t\mathbf{D}_t\mathbf{H}_t^T)\right)^T \mathbf{K_{YY}} \left(\prod_{t=1}^{s-1}(\mathbf{I} - \mathbf{H}_t\mathbf{D}_t\mathbf{H}_t^T)\right)\mathbf{H}_s\right]^{-1/2}$.
      - Evaluate $\mathbf{A}_s = \left[\left(\prod_{t=1}^{s-1}(\mathbf{I} - \mathbf{H}_t\mathbf{D}_t\mathbf{H}_t^T)\right)^T \mathbf{K_{YY}} \left(\prod_{t=1}^{s-1}(\mathbf{I} - \mathbf{H}_t\mathbf{D}_t\mathbf{H}_t^T)\right)\right]\mathbf{H}_s\mathbf{D}_s$.
    end
    - Update $\mathbf{Z}_s$ by identifying the index of absolute maximum in each row of $\mathbf{A}_s$.
  end
end

we evaluate the ensemble kernel matrix $\mathbf{K_{xY}} \in \mathbb{R}^{1\times N}$ where

$$\mathbf{K_{xY}}(i) = \mathcal{K}(\mathbf{x}, \mathbf{y}_i) = \sum_{r=1}^{R} \beta_r \mathcal{K}_r(\mathbf{x}_r, \mathbf{y}_{i,r}). \tag{6.36}$$

The weights, $\{\beta_r\}_{r=1}^R$, used for computing the ensemble kernel were obtained empirically during training. In order to perform a sparse approximation of the test sample using the kernel MLD, we compute a sparse coefficient for each level using the dictionary atoms from that level. Similar to the training procedure, we first compute the correlations between the test sample and all dictionary elements in level 1 as

$$\boldsymbol{\alpha}_1 = \Phi(\mathbf{x})^T\Phi(\boldsymbol{\Psi}_1) = \Phi(\mathbf{x})^T\Phi(\mathbf{Y}_1)\mathbf{H}_1\mathbf{D}_1 \tag{6.37}$$

$$= \mathbf{K_{xY}}\mathbf{H}_1\mathbf{D}_1. \tag{6.38}$$

Following this, we determine the $1 \times K$ membership vector $\mathbf{z}_1 = g(\boldsymbol{\alpha}_1)$ and the coefficient vector $\mathbf{h}_1 = \mathbf{z}_1 \odot \boldsymbol{\alpha}_1$. The residual vector of the test sample can be computed as

$$\Phi(\mathbf{r}_1) = \Phi(\mathbf{x}) - \Phi(\boldsymbol{\Psi}_1)\mathbf{h}_1^T, \tag{6.39}$$

$$= \Phi(\mathbf{x}) - \Phi(\mathbf{Y}_1)\mathbf{H}_1\mathbf{D}_1\mathbf{h}_1^T. \tag{6.40}$$

To determine a sparse coefficient in level 2, the residual vector $\mathbf{r}_1$ needs to be correlated with the dictionary atoms $\Phi(\boldsymbol{\Psi}_2)$. Generalizing this to any level s, we can evaluate the correlations between the residual $\Phi(\mathbf{r}_{s-1})$ and the dictionary atoms $\Phi(\boldsymbol{\Psi}_s)$ as

$$\boldsymbol{\alpha}_s = \mathbf{M}_s\mathbf{H}_s\mathbf{D}_s, \tag{6.41}$$

where

$$\mathbf{M}_s = \left[\mathbf{K}_{\mathbf{xY}} - \sum_{t=1}^{s-1} \mathbf{h}_t\mathbf{D}_t\mathbf{H}_t^T \left(\prod_{p=1}^{t-1}(\mathbf{I} - \mathbf{H}_p\mathbf{D}_p\mathbf{H}_p^T)\right)^T \mathbf{K}_{\mathbf{YY}}\right]\left[\left(\prod_{t=1}^{s-1}(\mathbf{I} - \mathbf{H}_t\mathbf{D}_t\mathbf{H}_t^T)\right)\right]. \tag{6.42}$$

The coefficient vector corresponding to level s can then be obtained as $\mathbf{h}_s = \mathbf{z}_s \odot \boldsymbol{\alpha}_s$, where $\mathbf{z}_s = g(\boldsymbol{\alpha}_s)$. The overall sparse code for the test sample $\mathbf{x}$ can be constructed by stacking $\mathbf{h}_s$'s from all levels, $\mathbf{h} = [\mathbf{h}_1\mathbf{h}_2 \ldots \mathbf{h}_S]^T$.

Note that, the complexity of evaluating sparse codes for a test sample with *Method 1* and *Method 2* cannot be compared directly since *Method 1* uses ℓ_1 minimization for obtaining sparse codes, and *Method 2* uses a customized multilevel procedure. Therefore, we compare the dominant complexity for obtaining ensemble kernel matrices in both the approaches. For *Method 1* the complexity of computing an ensemble kernel matrix $\mathbf{K}_{\boldsymbol{\Psi}\mathbf{x}}$, of $O(MK)$. Whereas, *Method 2* requires the computation of $\mathbf{K}_{\mathbf{xY}}$ and hence incurs a higher complexity of $O(MN)$, since typically $N > K$. However, as mentioned before, *Method 2* has the important advantage that

it does not place any restriction on the choice of the kernel function or the distance function for building the kernel matrix.

6.6 Object Recognition and Unsupervised Clustering

In this section, we describe the set of image descriptors and the corresponding kernel functions considered for our simulations and present discussions on the recognition performance using the Caltech-101 and Caltech-256 datasets. The same set of features were used to compute sparse codes that were subsequently used to obtain coefficient graphs for clustering.

6.6.1 Image Descriptors and Kernels

SIFT-ScSPM: For a given image, we extracted SIFT features [170] with three scales on a dense grid and used a K-means dictionary to obtain sparse codes for the local features. The number of dictionary elements was fixed at 1024. For each image, we generated the ScSPM feature using the algorithm in [100] and aggregated sparse codes using max-pooling at spatial scales 1, 2 and 4 respectively. The kernel matrix was constructed based on Euclidean distance between the features.

SS-ScSPM: The base kernel was constructed in the same way as the SIFT-ScSPM, except that the local SIFT descriptor was replaced by the self-similarity descriptor [171]. The size of the patch and the radius of the window were fixed at 5×5 and 40 respectively.

LBP-ScSPM: Another image descriptor was constructed using the ScSPM procedure, for Local Binary Pattern (LBP) [172] features extracted from overlapping patches in an image.

Gist: The images were resized to 128×128 and the gist descriptor was extracted from each image [173]. The kernel matrix was constructed based on Euclidean distance between the features.

PHOG: We extracted the PHOG descriptor from each image using the procedure

described in [174], by fixing the number of pyramid levels at 2. We used the Euclidean distance, and the χ^2 distance between the features for *Method 1* and *Method 2* respectively.

Biologically inspired features (C2-SWP, C2-ML): Biologically inspired C2 features proposed in [175] and [176] aim to mimic the simple and complex features in the human visual system. We extracted C2-SWP and C2-ML features, and used the Euclidean distance for both cases.

Geometric blur: For each image, we randomly sampled 400 edge pixels and applied the geometric blur descriptor [177] to them. We used the distance function proposed in [169] with these descriptors. Note that, because of the form of the distance function, this feature cannot be used in a fixed point dictionary update scheme. Therefore, when using this feature with *Method 1*, we use the initial dictionary obtained in the input space and refrain from updating it in the kernel space.

In order the initialize the algorithms to obtain MKSRs, the parameters in the descriptors and the distance functions were tuned independently. The criteria for tuning is that the resulting sparse codes with the base kernels individually achieved their best performances in classification using a linear SVM. When optimizing dictionaries and computing MKSRs, the ensemble kernel weights $\{\beta_r\}_{r=1}^{R}$ need to be empirically tuned, again to ensure high classification accuracies with a test set. To achieve this, we repeated the algorithms by the randomly splitting the data samples into train and test sets and determined the weights using cross-validation. We used the MATLAB interface of LIBLINEAR, a fast implementation of linear SVM for all our simulations [178]. The performance metric used is the percentage classification accuracy.

In addition to object recognition, spectral clustering can be performed using the graph created from the kernel sparse codes. Using the coefficient matrix $\mathbf{A}$, we constructed the non-negative graph weight matrix $\mathbf{W} = |\mathbf{A}^T\mathbf{A}|$. The normalized

Table 6.2: Comparison of the classification accuracies on the Caltech-101 dataset. For the proposed algorithms, results were obtained by averaging over 10 different train and test datasets chosen at random.

Method	**# Training samples per class**					
	5	**10**	**15**	**20**	**25**	**30**
Zhang *et.al.* [169]	46.6	55.8	59.1	62	-	66.2
Lazebnik *et.al.* [99]	-	-	56.4	-	-	64.6
Griffin *et.al.* [179]	44.2	54.5	59	63.3	65.8	67.6
Jain *et.al.* [180]	-		65	-	-	70.4
Boiman *et.al.* [181]	-	-	61	-	-	69.1
Pham *et.al.* [182]	-	-	42	-	-	-
Gemert *et.al.* [183]	-	-	-	-	-	64.16
Yang *et.al.* [100]	-	-	67	-	-	73.2
Wang *et.al.* [154]	51.15	59.77	65.43	67.74	70.16	73.44
Aharon *et.al.* [83]	49.8	59.8	65.2	68.7	71	73.2
Zhang *et.al.* [157]	49.6	59.5	65.1	68.6	71.1	73
Jiang *et.al.* [152]	54	63.1	67.7	70.5	72.3	73.6
MKSR (Method 1)	**58.34**	**66.81**	**70.83**	**74.02**	**76.1**	**77.8**
MKSR (Method 2)	**58.9**	**67.3**	**71.44**	**74.7**	**76.83**	**78.01**

weight matrix is given as $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, where $\mathbf{D}$ is a diagonal matrix with the i^{th} diagonal element containing the sum of the all the elements in the i^{th} row or column of $\mathbf{W}$. Assuming that there are C clusters, the eigenvectors corresponding to the C largest eigenvalues are stacked in the matrix $\mathbf{U} = [\mathbf{u}_1\mathbf{u}_2 \dots \mathbf{u}_C]$. The rows of the matrix are then clustered to obtain the cluster labels. The standard metrics, the accuracy and the normalized mutual information, are used to quantify the clustering performance [34].

6.6.2 Simulation Results

Caltech-101

The Caltech-101 dataset [184] consists of 9144 images belonging to 101 object categories and an additional class of background images. The number of images in each category varies roughly between 40 and 800. Although the target object often appears in the central region of the images, the large number of categories and the intraclass

variations makes this dataset challenging. This dataset has been often used as a benchmark for evaluating the performance of supervised classifiers. Note that, we resized all images to be no larger than 300×300 with the aspect ratio preserved. Following the common evaluation procedure, we trained the classifiers on $5, 10, 15, 20, 25$ and 30 images per class and evaluated the performance by testing on the rest.

For the proposed *Method 1*, we extracted the different image descriptors and constructed the corresponding base kernels as described in Section 6.6.1. As discussed in Section 6.4, we need to obtain separate dictionaries for each descriptor in the original input space. Hence, we generated a K-means dictionary containing 2048 elements for each descriptor set to perform sparse coding of the descriptors. Following this, we computed the ensemble kernel matrices using the expressions in (6.15), (6.16) and (6.17) respectively. The parameter γ for computing the kernel sparse codes using the Feature-Sign search algorithm was fixed at 0.3. We also performed kernel dictionary learning using the proposed *Method 2* described in Section 6.5.2 fixing the number of levels at 16 and the number of atoms per level at 128, resulting in a total of 2048 atoms.

The quantitative results of the proposed multiple kernel sparse representations frameworks are presented in Table 6.2. The recognition rates presented are averaged over 10 iterations with train and test datasets chosen at random. As it can be observed, the proposed approaches achieve higher classification rates in comparison to other sparse coding based approaches. For example, consider the case when the number of training samples $N_{train} = 15$. Lazebnik *et.al.* [99] reported a recognition rate of 56.4 percent by considering the spatial pyramid matching kernel. In [169], Zhang *et.al.* combined the geometric blur descriptor with the spatial information to achieve an improved recognition rate of 59.1 percent. By replacing the bag-of-words procedure in SPM by sparse coding, the ScSPM algorithm proposed in [100] improves the classification performance to 67 percent. The LC-KSVD algorithm developed

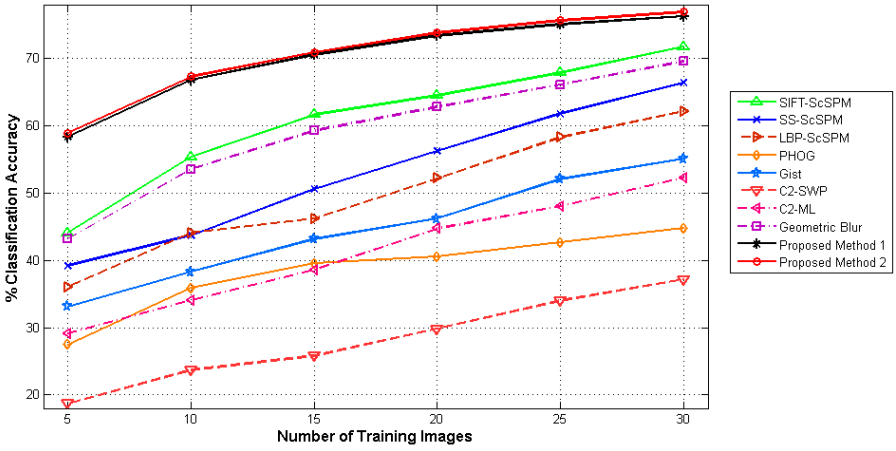


Figure 6.3: Classification performance obtained by using each base kernel separately with *Method 2* on the Caltech-101 dataset. In each case, we report the results obtained by using different number of training images per class. For comparison, we show the classification accuracies achieved with multiple kernels using the two proposed algorithms.

in [152], performed label consistent coding to further improve the recognition to 67.7 percent. By incorporating multiple kernels into the sparse coding procedure, we obtained the classification accuracies of 70.56 and 70.94 percent using the proposed *Method 1* and *Method 2* respectively.

In order to demonstrate the importance of fusing the multiple kernels, we performed object recognition by considering each base kernel separately. In each case, we tuned the kernel parameter to yield the best recognition performance with the proposed *Method 2*. Figure 6.3 illustrates the classification accuracies obtained, with each base kernel, for different number of training images per class. For comparison, we show the accuracies obtained for the two proposed multiple kernel methods as well. The improvement in recognition by using multiple kernels is apparent in all cases. For example, when $N_{train} = 15$ the SIFT-ScSPM and the geometric blur descriptors provide the best accuracies of 61.65% and 59.29% respectively, while the C2-SWP descriptor achieves a very low recognition rate of 25.8%. However, when all

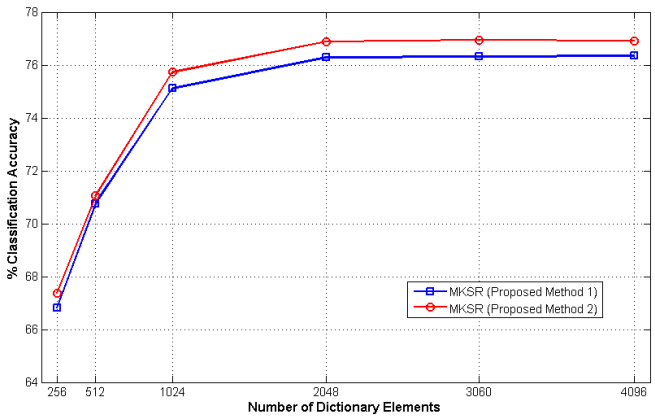


Figure 6.4: Classification accuracies of the proposed MKSR algorithms on the Caltech-101 dataset using dictionaries of different sizes. In each case, we report the mean accuracy obtained by using 30 images per class for training, for 10 different train and test sets chosen at random.

the kernels are combined using the proposed methods we achieve improvements of 8.91% and 9.29% in the mean recognition performance, when compared to using just SIFT-ScSPM descriptors.

Finally, we demonstrate the effect of dictionary size on the classification performance for both the proposed methods. We varied the number of dictionary elements between 256 and 4096 and repeated the simulations with 30 training samples per class, using 10 different random train and test sets in each case. Figure 6.4 plots the mean percentage classification accuracy for different dictionary sizes. We observed that beyond $K = 1024$, the classification rate does not improve significantly with the size of the dictionary.

As described earlier we can construct graphs, with the multiple kernel sparse codes obtained using the proposed algorithms, for spectral clustering. We evaluate the clustering performance of these graphs using a benchmark subset of the Caltech-101 dataset [164]. Images from the following 20 classes were used to learn dictionaries and obtain multiple kernel sparse codes: faces, motorbikes, dollar bill, GarïňĄeld, Snoopy, stop sign, windsor chair, leopards, binocular, brain, camera, car side, ferry,

Table 6.3: Comparison of the clustering performance obtained using graphs, constructed from the kernel sparse codes, on a subset of Caltech-101. In each case, the results were obtained by averaging over 50 trials.

Feature	**% Accuracy**		**NMI**	
	Ave.	**Max.**	**Ave.**	**Max.**
SIFT-ScSPM	60.6	61.3	0.63	0.66
SS-ScSPM	52.8	54.31	0.52	0.59
PHOG	45.4	46.1	0.47	0.51
Gist	44.6	46.9	0.41	0.48
C2-SWP	38.31	39.6	0.33	0.36
C2-ML	31.8	33.1	0.29	0.35
GB	49.6	50.2	0.5	0.53
MKSR (Method 1)	**69.8**	**74.7**	**0.74**	**0.81**
MKSR (Method 2)	**70.4**	**75.13**	**0.79**	**0.83**

Table 6.4: Comparison of the classification accuracies on the Caltech-256 dataset. For the two proposed algorithms, the reported results were obtained by averaging over 10 different train and test datasets chosen at random.

Method	**# Training samples per class**			
	15	**30**	**45**	**60**
Gemert *et.al.* [183]	-	27.17	-	-
Griffin *et.al.* [179]	28.3	34.1	-	-
Yang *et.al.* [100]	27.73	34.02	37.46	40.14
Guo *et.al.* [167]	29.77	35.67	38.61	40.3
Wang *et.al.* [154]	34.46	41.19	45.31	47.68
MKSR (Method 1)	**38.72**	**45.12**	**48.65**	**50.27**
MKSR (Method 2)	**39.22**	**45.61**	**49.02**	**50.51**

hedgehog, pagoda, rhino, stapler, water Lilly, wrench, and yin yang. Similar to the object recognition simulations, we constructed graphs using kernel sparse codes of the individual features for comparison. The dictionary size was fixed at 2048 for all cases. Table 6.3 shows the clustering accuracy and normalized mutual information obtained using the different graphs. In each case, both the average and maximum values obtained over 50 trials are reported. As it can be observed, the ensemble features perform significantly better than the individual features similar to the object recognition simulation.

Caltech-256

The Caltech-256 dataset [179] contains $30,607$ images in 256 categories and its variability makes it extremely challenging in comparison to the Caltech-101 dataset. The intraclass variance is quite high and the objects of interest are not necessarily in the center of the images. The experimental setup is similar to the previous section and we evaluated the recognition performance with N_{train} fixed at $15, 30, 45$, and 60 images per class respectively. The number of dictionary elements are fixed at 4096 for both the algorithms. Table 6.4 shows the recognition rates obtained with the different methods and similar to the previous case, the proposed MKSR algorithms outperform other baseline methods.

Chapter 7

AUTOMATED TUMOR SEGMENTATION USING KERNEL SPARSE REPRESENTATIONS

7.1 Problem Statement

A robust method to automatically segment a medical image into its constituent heterogeneous regions can be an extremely valuable tool for clinical diagnosis and disease modeling. Given a reasonably large data set, performing manual segmentation is not a practical approach. Brain tumor detection and segmentation have been of interest to researchers over recent years and currently, there exists no comprehensive algorithm built and adopted in the clinical setting [185]. Although patient scans can be obtained using different imaging modalities, Magnetic Resonance Imaging (MRI) has been commonly preferred for brain imaging over other modalities because of its non-invasive and non-ionizing nature, and because it allows for direct multi-plane imaging.

Tumors may be malignant or benign as determined by a biopsy, and are known to affect brain symmetry and cause damage to the surrounding brain tissues. Automated tumor segmentation approaches are often challenged by the variability in size, shape and location of the tumor, the high degree of similarity in the pixel intensities between normal and abnormal brain tissue regions, and the intensity variations among identical tissues across volumes. As a result, unsupervised thresholding techniques have not been very successful in accurate tumor segmentation [186]. Furthermore, approaches that incorporate prior knowledge of the normal brain from atlases require accurate non-rigid registration [187], [188], and hence generating adequate segmentation results potentially calls for user-intervention and/or a patient specific training system. In addition, these methods require elaborate pre-processing and they tend to over-estimate the tumor volume.

Approaches for tumor segmentation can be either region-based or pixel-based. The active contours method [189] is a widely adopted region-based approach that is usually combined with a level-set evolution for convergence to a region of interest [190]. However, it is sensitive to the contour initialization, and has a high computational cost due to its iterative nature. Model-based approaches [191] employ geometric priors to extend the Expectation Maximization (EM) algorithm to augment statistical classification. In relatively homogeneous cases such as low grade gliomas, the outlier detection framework proposed by Prastawa *et al.* [186] [192] was shown to perform well.

Pixel-based approaches such as Fuzzy C-Means (FCM) using neighborhood labels [193], Conditional Random Fields [194], Bayesian model-aware affinities extending the SWA algorithm [185], and the more recent graph-based techniques combined with the Cellular-Automata (CA) algorithm [195] have also achieved some success in tumor segmentation. However, processing issues with respect to contour initialization, noise reduction, intensity standardization, cluster selection, spatial registration, and the need for accurate manual seed-selection leaves substantial room for improvement. In addition, building a robust automated approach that does not require user intervention is very important, particularly for processing large datasets.

7.1.1 Sparsity in Tumor Segmentation

Sparse models emulate the activity of neural receptors in the primary visual cortex of the human brain. In [18], Olshausen and Field demonstrated that learning sparse linear codes for natural images results in a family of localized, oriented, and bandpass features, similar to those found in the primary visual cortex. Sparsity of the coefficients has been exploited in a variety of signal, and image processing applications including compression [83], denoising [139], compressed sensing [196], source separation [121], face classification [96], and object recognition [153].

Despite its great applicability, the use of sparse models in complex visual recognition applications presents three main challenges: (i) linear generative model of sparse coding can be insufficient for modeling the non-linear relationship between the complex image features, (ii) in several visual recognition tasks, no single descriptor can efficiently model the whole data set, i.e., there is a need to integrate multiple image features into the sparse coding paradigm, and (iii) sparse models require data samples to be represented in the form of feature vectors, and it is not straightforward to extend them to the case of other forms such as pixel values, matrices or higher order tensors. In order to circumvent the aforementioned challenges, kernel learning methods can be incorporated in sparse coding [41]. The kernel methods map the data samples into a high-dimensional feature space, using a non-linear transformation, in which the relationship between the features can be represented using linear models. By ensuring that the resulting feature space is a Hilbert space, kernel methods can work by considering only the similarity between the features, and not the features themselves. By developing approaches for sparse coding and dictionary learning in the feature space, novel frameworks can be designed for computer vision tasks such as recognition and segmentation.

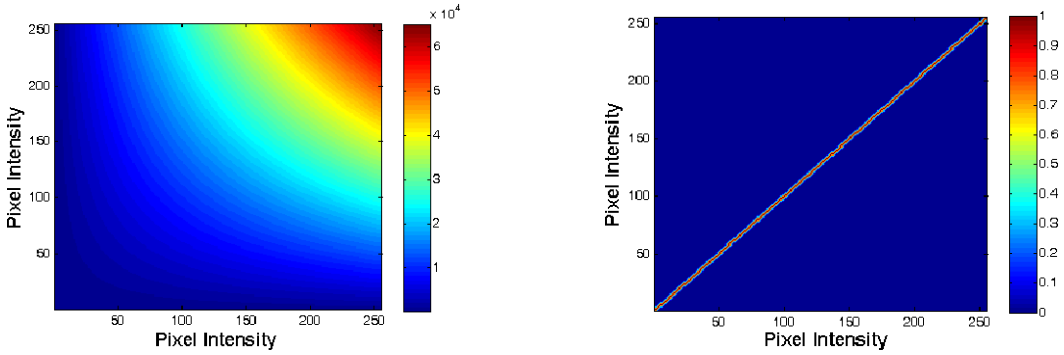
In this chapter, we develop a novel approach to automatically segment enhancing/active and necrotic tumor components from T1-weighted contrast-enhanced MR images. We propose to compute kernel sparse codes for the pixels in the image and perform pixel-based segmentation using those codes. Furthermore, we develop the kernel K-lines clustering algorithm to learn kernel dictionaries for coding the pixels. The proposed algorithm for localizing the active tumor regions uses an ensemble kernel constructed using pixel intensities and their spatial locations. Each pixel is classified as belonging to a tumor or a non-tumor region using a linear SVM on the kernel sparse codes. Finally, we propose a semi-automated segmentation technique for improved computational efficiency, wherein the user can initialize the tumor region.

This approach eliminates the need to incorporate the spatial location information and reduces the number of pixels to be processed. In addition, we show that the complex linear SVM classifier can be replaced by a simple error-based classifier without compromising the segmentation quality. We evaluate the proposed algorithm on a set of T1-weighted contrast-enhanced MR images and compare the results with manual segmentation performed by an expert radiologist.

7.2 Kernel Sparse Coding for Tumor Segmentation

Sparse coding algorithms are typically employed for vectorized patches or feature vectors extracted from the images, using an overcomplete dictionary. However, the proposed tumor identification algorithm aims to obtain sparse codes for the pixel values directly. This is trivial since $M = 1$ in this case. Furthermore, in order to discriminate between the pixels belonging to multiple segments, we may need to consider the *non-linear similarity* between them. Though the linear generative model of sparse coding has been effective in several image understanding problems, it does not consider the non-linear similarities between the training samples.

It is typical in machine learning methods to employ the *Kernel Trick* to learn linear models in a feature space that captures the non-linear similarities. The *Kernel Trick* maps the non-linear separable features into a feature space $\mathcal{F}$ using a transformation $\Phi(.)$, in which similar features are grouped together. By performing sparse coding in the feature space $\mathcal{F}$, we can obtain highly discriminative codes for samples from different classes [167]. Note that the choice of the non-linear transformation is crucial to ensure discrimination. The transformation $\Phi(.)$ is chosen such that $\mathcal{F}$ is a Hilbert space with the reproducing kernel $\mathcal{K}(.,.)$ and hence the non-linear similarity between two samples in $\mathcal{F}$ can be measured as $\mathcal{K}(\mathbf{y}_i, \mathbf{y}_j) = \Phi(\mathbf{y}_i)^T \Phi(\mathbf{y}_j)$. Note that the feature space is usually high-dimensional (sometimes infinite) and the closed form expression for the transformation $\Phi(.)$ may be intractable or unknown. There-



(a) (b)

Figure 7.1: Similarity between grayscale pixel intensities (0 to 255): (a) linear similarity ($y_i y_j$) and (b) non-linear similarity ($\mathcal{K}(y_i, y_j)$) using an RBF kernel.

fore, we simplify the computations by expressing them in terms of inner products $\Phi(\mathbf{y}_i)^T\Phi(\mathbf{y}_j)$, which can then be replaced using $\mathcal{K}(\mathbf{y}_i, \mathbf{y}_j)$, the value of which is always known. This is referred to as the *Kernel Trick*. Note that in order for a kernel to be valid, the kernel function or the kernel matrix should be symmetric positive semidefinite according to Mercer's theorem [197].

In this chapter, we use the Radial Basis Function (RBF) kernel of the form $\mathcal{K}(y_i, y_j) = \exp(-\gamma(y_i - y_j)^2)$, which leads to discriminative sparse codes. As a simple demonstration, the difference between linear similarity of grayscale pixel intensities (0 to 255) and the non-linear similarities obtained using the RBF kernel ($\gamma = 0.3$) is illustrated in Figure 7.1(a) and (b). The linear similarities depend predominantly on the individual intensities of the pixels and not on the closeness of intensities. Whereas, when the RBF kernel is used, the pixel intensities that are close to each other have high non-linear similarity irrespective of the intensities. Pixels with intensities that are far apart have zero non-linear similarity. Therefore, the pixelwise sparse codes that we obtain using such a kernel will behave similarly.

7.2.1 Kernel Sparse Coding

Given the feature mapping function $\Phi : \mathbb{R}^M \mapsto \mathbf{R}^G$, the generative model in $\mathcal{F}$ for kernel sparse coding is given by $\Phi(\mathbf{y}) = \Phi(\mathbf{D})\mathbf{x} + \mathbf{n}$. We denote the data sample $\mathbf{y}$ in the feature space as $\Phi(\mathbf{y})$ and the dictionary by $\Phi(\mathbf{D}) = [\Phi(\mathbf{d}_1), \Phi(\mathbf{d}_2), ..., \Phi(\mathbf{d}_K)]$. The kernel similarities $\mathcal{K}(\mathbf{y}_i, \mathbf{y}_j) = \Phi(\mathbf{y}_i)^T\Phi(\mathbf{y}_j)$, $\mathcal{K}(\mathbf{d}_k, \mathbf{y}) = \Phi(\mathbf{d}_k)^T\Phi(\mathbf{y})$ and $\mathcal{K}(\mathbf{d}_k, \mathbf{d}_l) = \Phi(\mathbf{d}_k)^T\Phi(\mathbf{d}_l)$ can be computed using pre-defined kernel functions (RBF in our case). All further computations in the feature space should be performed exclusively using kernel similarities. The problem of sparse coding can be posed in the feature space as

$$\min_{\mathbf{x}} \|\Phi(\mathbf{y}) - \Phi(\mathbf{D})\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1. \tag{7.1}$$

Expanding the objective in (7.1) we obtain

$$\begin{aligned}
&\Phi(\mathbf{y})^T\Phi(\mathbf{y}) - 2\mathbf{x}^T\Phi(\mathbf{D})^T\Phi(\mathbf{y}) + \mathbf{x}^T\Phi(\mathbf{D})^T\Phi(\mathbf{D})\mathbf{x} + \lambda\|\mathbf{x}\|_1, \\
&= \mathbf{K_{yy}} - 2\mathbf{x}^T\mathbf{K_{Dy}} + \mathbf{x}^T\mathbf{K_{DD}}\mathbf{x} + \lambda\|\mathbf{x}\|_1, &(7.2)\\
&= F(\mathbf{x}) + \lambda\|\mathbf{x}\|_1. &(7.3)
\end{aligned}$$

Here, $\mathbf{K_{yy}}$ is the element $\mathcal{K}(\mathbf{y}, \mathbf{y})$, $\mathbf{K_{Dy}}$ is a $K \times 1$ vector containing the elements $\mathcal{K}(\mathbf{d}_k, \mathbf{y})$, $\forall k = \{1, \ldots, K\}$ and $\mathbf{K_{DD}}$ is a $K \times K$ matrix containing the kernel similarities between the dictionary atoms. Clearly, the modified objective function is similar to the sparse coding problem, except for the use of the kernel similarities. Hence, the kernel sparse coding problem can be efficiently solved using the feature-sign search algorithm or LARS. However, it is important to note that the computation of kernel matrices incurs additional complexity. Since the dictionary is fixed in (7.3), $\mathbf{K_{DD}}$ is computed only once and the complexity of computing $\mathbf{K_{Dy}}$ grows as $O(MK)$.

7.3 Kernel Dictionary Design

Optimization of dictionaries in the feature space can be carried out by reposing the dictionary learning procedures using only the kernel similarities. Such non-linear

dictionaries can be effective in yielding compact representations, when compared to approaches such as the kernel PCA, and in modeling the non-linearity present in the training samples. In this section, we will describe the formulation of a kernel dictionary learning procedure, and demonstrate its effectiveness in representation and discrimination.

In order to design the dictionary $\Phi(\mathbf{D})$, we will adapt dictionary learning to the feature space, with the constraint that only one element in the sparse code can be non-zero. This is a special case of the kernel dictionary learning proposed in [198]. This procedure is equivalent to the kernel version of K-lines clustering, which attempts to fit K 1-D subspaces to the training data in $\mathcal{F}$ [35]. Though sophisticated kernel dictionaries can be designed, employing dictionaries obtained using this clustering procedure results in good performance for our tumor segmentation problem. The clustering procedure can be solved using

$$\min_{\mathbf{A},\mathbf{X}} \|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2 \text{ such that } \|\mathbf{x}_i\|_0 \leq 1, \forall i. \tag{7.4}$$

Each dictionary atom $\Phi(\mathbf{d}_i)$ corresponds to a cluster center and each coefficient vector $\mathbf{x}_i$ encodes the cluster association as well as the weight corresponding to the i^{th} pixel. Let us define K membership sets $\{\mathcal{C}_k\}_{k=1}^K$, where $\mathcal{C}_k$ contains the indices of all training vectors that belong to the cluster k. The alternating optimization for solving (7.4) consists of two steps: (a) cluster assignment, which involves finding the association and weight of each training vector and hence updating the sets $\{\mathcal{C}_k\}_{k=1}^K$, and (b) cluster update, which involves updating the cluster center by finding the centroid of training vectors corresponding to each set $\mathcal{C}_k$.

In the cluster assignment step, we compute the correlations of a training sample, with the dictionary atoms as $\Phi(y_i)^T\Phi(\mathbf{D}) = \mathbf{K}_{y_i\mathbf{Y}}\mathbf{A}$. If the k^{th} dictionary atom results in maximum absolute correlation, the index i is placed in set $\mathcal{C}_k$, and the corresponding non-zero coefficient is the correlation value itself. For the cluster k, let

$\Phi(\mathbf{Y}_k) = \Phi(\mathbf{Y})\mathbf{E}_k$ be the set of member vectors and $\mathbf{x}_k^R$ be the row of corresponding non-zero weights. The cluster update involves solving

$$\min_{\mathbf{a}_k} \|\Phi(\mathbf{Y})\mathbf{a}_k\mathbf{x}_k^R - \Phi(\mathbf{Y})\mathbf{E}_k\|_F^2. \tag{7.5}$$

Denoting the singular value decomposition of

$$\Phi(\mathbf{Y}_k) = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T, \tag{7.6}$$

the rank-1 approximation, which also results in the optimal solution for (7.5), is given by

$$\Phi(\mathbf{Y})\mathbf{a}_k\mathbf{x}_k^R = \mathbf{u}_{k1}\sigma_{k1}\mathbf{v}_{k1}^T, \tag{7.7}$$

where σ_{k1} is the largest singular value, and $\mathbf{u}_{k1}$ and $\mathbf{v}_{k1}$ are the columns of $\mathbf{U}_k$ and $\mathbf{V}_k$ corresponding to that singular value. Eqn. (7.7) implies that $\Phi(\mathbf{Y})\mathbf{a}_k = \mathbf{u}_{k1}$ and $\mathbf{x}_k^R = \sigma_{k1}\mathbf{v}_{k1}^T$. Let the eigen decomposition of $\mathbf{K}_{\mathbf{Y}_k\mathbf{Y}_k}$ be $\mathbf{V}_k\mathbf{\Delta}_k\mathbf{V}_k^T$ and hence we have $\sigma_{k1} = \sqrt{\Delta_k(1,1)}$, assuming the eigen values are in descending order. From (7.6), we also have $\Phi(\mathbf{Y}_k)\mathbf{v}_{k1} = \sigma_{k1}\mathbf{u}_{k1}$. Substituting for $\Phi(\mathbf{Y}_k)$ and $\mathbf{u}_{k1}$, we obtain $\Phi(\mathbf{Y})\mathbf{E}_k\mathbf{v}_{k1} = \sigma_{k1}\Phi(\mathbf{Y})\mathbf{a}_k$, which results in

$$\mathbf{a}_k = \sigma_{k1}^{-1}\mathbf{E}_k\mathbf{v}_{k1}. \tag{7.8}$$

Note that $\mathbf{a}_k$ completely defines $\mathbf{d}_k$. The cluster assignment and update steps are repeated until convergence, i.e., when $\{\mathcal{C}_k\}_{k=1}^K$ does not change over iterations.

7.3.1 Representation

Kernel sparse coding can be used as an alternative to approaches such as kernel PCA for efficient data representation. Though complete reconstruction of the underlying data from the kernel sparse codes requires computation of pre-images [199], novel test samples can be well approximated using the learned kernel dictionaries. As a demonstration, we consider the class of digit 2 from the USPS dataset and use a subset of images for training a kernel dictionary using kernel K-lines clustering.

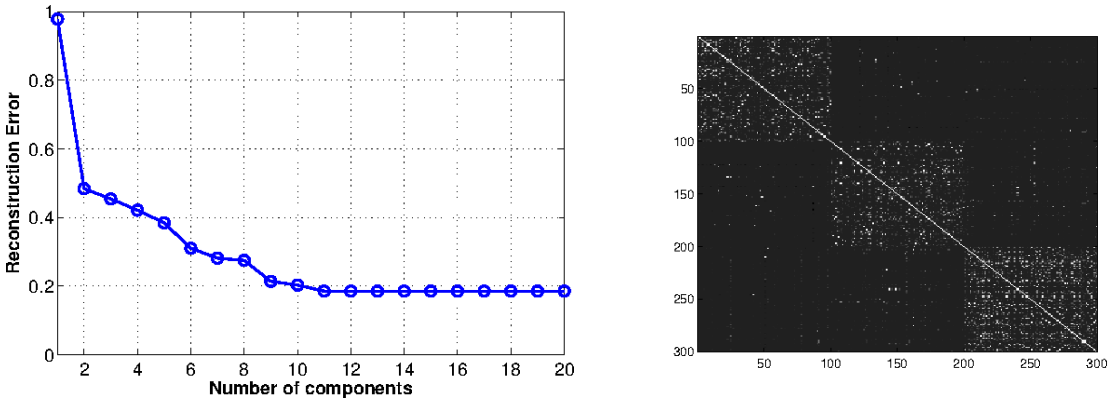


Figure 7.2: (a) Reconstruction error for a novel test sample using kernel sparse coding, for different values of sparsity. (b) Similarity between the kernel sparse codes of samples drawn from 3 different classes in the USPS dataset. Since the kernel codes of samples belonging to the same class are highly similar, we observe a block-wise structure in the normalized correlation plot.

We then compute sparse code for a novel test sample $\mathbf{z}$, different from the training set, and compute the reconstruction error as $\|\Phi(\mathbf{z}) - \Phi(\mathbf{D})\mathbf{a}\|_2^2$. Figure 7.2(a) shows the reconstruction error obtained for a test sample for different values of sparsity, $\{1, \ldots, 20\}$.

7.3.2 Discrimination

In addition to efficiently modeling data samples, kernel sparse coding is well suited for supervised learning tasks. Since the non-linear similarities between the training samples are considered while learning the dictionary, the resulting codes are highly discriminative. As a demonstration, we consider 100 training samples each from 3 different classes in the USPS dataset (Digits $3, 4$ and 7). We obtain the kernel sparse codes for all the samples and compute the normalized cross correlation between the sparse features. In cases of high discrimination, we expect the features belonging to a class to be highly similar to each other compared to samples from other classes. The block-wise structure in the normalized correlation plot in Figure 7.2(b) evidences the discrimination power of the kernel sparse codes.

7.4 Proposed Automated Tumor Segmentation Algorithm

The proposed algorithm employs a pixel-based approach to determine tumor regions in the MR image. In order to determine if a pixel belongs to a tumor region, adaptive thresholding techniques can be used. However, building more sophisticated tools can improve segmentation performance. In this section, we describe the proposed algorithm for automated tumor segmentation based on kernel sparse codes.

To perform tumor segmentation, we need to identify pixels that can possibly constitute a tumor region based on intensity. Though segmentation is as an unsupervised learning problem, we can pose it is as a supervised learning problem since we can easily obtain a at least a few training images with tumor regions marked by an expert. Hence, we propose to obtain kernel dictionaries using the training samples and learn a 2-class classifier (Tumor vs Non-tumor). Furthermore, in order to localize the tumor regions in the image, we need to incorporate additional constraints to ensure connectedness among pixels in a segment. This can be addressed by building a spatial location kernel and fusing it with the intensity kernel.

7.4.1 Combining Multiple Features

The use of multiple features to more precisely characterize images has been a very successful approach for several classification tasks. Though this method provides the flexibility of choosing features to describe different aspects of the underlying data, the resulting representations are high-dimensional and the descriptors can be very diverse. Hence, there is a need to transform the features to a unified space that facilitates the recognition tasks, and construct low dimensional compact representations for the images in the unified space.

Let us assume that a set of R diverse descriptors are extracted from a given image. Since the kernel similarities can be used to fuse the multiple descriptors, we need to build the base kernel matrix for each descriptor. Given a suitable distance

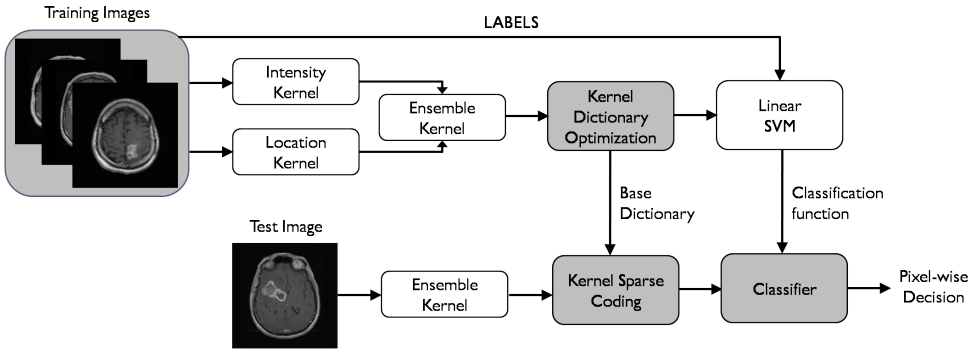


Figure 7.3: Illustration of the proposed algorithm for automated tumor segmentation. For a set of training samples, the ensemble kernel dictionary is obtained using Kernel K-lines clustering procedure, and a 2-class linear SVM is used to classify the pixels.

function d_r, which measures the distance between two samples for the feature r, we can construct the kernel matrix as

$$\mathbf{K}_r(i,j) = \mathcal{K}_r(\mathbf{y}_i, \mathbf{y}_j) = \exp(-\gamma d_r^2(\mathbf{y}_i, \mathbf{y}_j)), \tag{7.9}$$

where γ is a positive constant. Given the R base kernel matrices, $\{\mathbf{K}_r\}_{r=1}^R$, we can construct the ensemble kernel matrix as

$$\mathbf{K} = \sum_{r=1}^{R} \beta_r \mathbf{K}_r, \quad \forall \beta_r \geq 0. \tag{7.10}$$

Note that the ensemble matrix can be constructed in other ways also. Alternatively, the descriptors can be fused as

$$\mathbf{K} = \mathbf{K}_1 \odot \mathbf{K}_2 \odot \ldots \odot \mathbf{K}_R, \tag{7.11}$$

where $\odot$ denotes the Hadamard product between two matrices. Performing sparse coding using the ensemble kernel matrices will take the R features into account. Note that when combining kernel matrices we need to ensure that the resulting kernel matrix also satisfies the Mercer's conditions.

7.4.2 Algorithm

The proposed algorithm for automated tumor segmentation is illustrated in Figure 7.3. In the rest of this chapter, we refer to this as the *Kernel Sparse Coding-based*

Automated (KSCA) segmentation algorithm. In the training stage, it is assumed that the location of the tumor pixels are known in the ground truth training images. For a subset of T pixels (both positive and negative examples) obtained from the training images, we compute the intensity kernel matrix, $\mathbf{K}_I \in \mathbb{R}^{T \times T}$, by employing an RBF kernel on the pixel intensity values. In addition, the spatial location kernel matrix $\mathbf{K}_L$ is constructed as

$$\mathbf{K}_L(i,j) = \mathcal{K}_L(y_i, y_j) = \begin{cases} \exp^{-\gamma \|\mathbf{L}_i - \mathbf{L}_j\|_2^2}, & \text{if } \ j \in \mathcal{N}(i), \\ 0, & \text{otherwise.} \end{cases} \tag{7.12}$$

Here, $\mathcal{N}(i)$ denotes the neighborhood of the pixel y_i, and $\mathbf{L}_i$ and $\mathbf{L}_j$ are the locations of the pixels, y_i and y_j respectively. We fuse the intensity and spatial location kernel matrices to obtain the ensemble kernel matrix, $\mathbf{K} = \mathbf{K}_I \odot \mathbf{K}_L$.

The sparse codes obtained with a dictionary learned in the ensemble feature space model the similarities with respect to both intensity and location of pixels. A set of training images, with active tumor regions, are used to learn a kernel dictionary with the kernel K-lines clustering procedure. Using the kernel sparse codes belonging to tumor and non-tumor regions, we learn 2-class linear SVM to classify the pixel. For a test image, we obtain the required ensemble kernel matrices and compute the kernel sparse codes using the learned dictionary. Finally, the SVM classifier can be used to identify the pixels belonging to an active tumor region. The impact of combining diverse features using kernel sparse coding is evidenced by the accurate segmentation results.

7.5 Complexity Reduction using a Semi-Automated Approach

The amount of training required and the computational complexity are two important factors that can determine the efficiency of an automated segmentation algorithm. Since the dictionary training is performed using pixels, the number of training images used is quite limited. Though the computational complexity of the automated

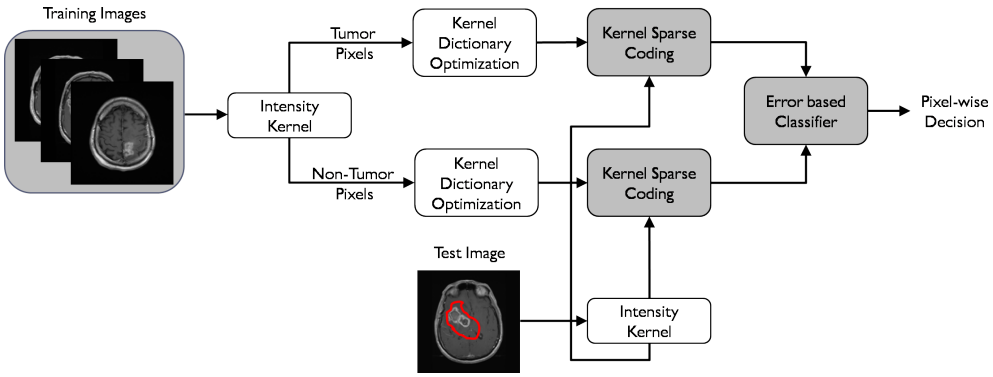


Figure 7.4: Illustration of the approach for complexity reduction in the proposed algorithm. By allowing the user to initialize the tumor region in a test image, the need for incorporating locality information is eliminated. Furthermore, the SVM classifier can be replaced by a simple reconstruction error-based classifier.

segmentation algorithm described earlier is comparable to several existing methods, its efficiency can be further improved by allowing the user to initialize the tumor region. Computing the kernel sparse codes for all pixels in a test image incurs the maximum complexity and hence initializing the tumor regions drastically reduces the number of pixels to be processed. Furthermore, there is no need to explicitly include the location information in the algorithm, since the tumor region has already been localized by the user. Hence, the classification can be carried out by using a simple error-based classifier on the kernel sparse codes. We refer to this as the *Kernel Sparse Coding-based Semi-Automated* (KSCSA) segmentation approach. We observed from our experiments that for an average sized tumor region, we achieve significant speedup by using the semi-automated approach. However, the segmentations obtained using the two methods are quite comparable, though the automated approach can potentially generate more false positives when compared to the semi-automated approach.

Given a set of training images containing active tumor regions, we use the tumor and non-tumor pixels to train two separate kernel dictionaries. We construct two RBF kernel matrices on the pixel intensities and employ the kernel K-lines clustering algorithm to learn the tumor and non-tumor dictionaries, $\Phi(\mathbf{D}_T)$ and $\Phi(\mathbf{D}_N)$,

respectively. Note that dictionary learning is performed only once, and as we will show in our experimental results, the dictionaries generalize well to reasonably large datasets.

For a test image, we obtain kernel sparse codes for each pixel y_i using $\Phi(\mathbf{D}_T)$ and $\Phi(\mathbf{D}_N)$, and denote the respective sparse codes as $\mathbf{x}_i^T$ and $\mathbf{x}_i^N$. Since the dictionaries are optimized for two different classes of pixel intensities, we expect the tumor pixels to be better modeled by the tumor dictionary. Hence we classify a pixel as belonging to an active tumor region if the approximation error obtained with the tumor dictionary is less than that obtained with the non-tumor dictionary:

$$\mathcal{J}(y_i) = \begin{cases} \textit{Tumor}, & \text{if } E_N - E_T \geq \epsilon, \\ \textit{Non-tumor}, & \text{otherwise.} \end{cases} \tag{7.13}$$

Here the approximation errors with respect to the two dictionaries are $E_N = \|\Phi(y_i) - \Phi(\mathbf{D}_N)\mathbf{x}_i^N\|_2$ and $E_T = \|\Phi(y_i) - \Phi(\mathbf{D}_T)\mathbf{x}_i^T\|_2$, respectively. Note that the threshold for the error difference, ϵ, can be tuned using a validation dataset before applying the algorithm to the test data.

7.6 Experiments

In this section, we provide details about the datasets used to evaluate our algorithm and present the segmentation results. The results are compared to manual segmentations performed by a radio-oncology specialist, based on both the subjective visual quality and quantitative standards such as *Accuracy (Acc)* and *Correspondence Ratio (CR)*.

7.6.1 Dataset

The algorithm was tested on a set of T1-weighted (spin echo) contrast-enhanced, 2-D Dicom format images acquired with a 1.5T GE Genesis Signa MR scanner. Each axial slice was 5 mm thick with a 7.5 mm gap between slices, and the size of the image matrix was 256×256. Patients were administered a 20cc Bolus of Gadolinum

contrast agent, and were already diagnosed with Glioblastoma Multiforme (GBM), the most common and dangerous malignant primary brain tumor. These tumors are characterized by jagged boundaries with a ring enhancement, possibly a dark core necrotic component, and are accompanied by edema (swelling). The ground truth (GT) images were obtained from the manual segmentation carried out by an expert radiologist at the St. Joseph's Hospital and Medical Center in Phoenix. We tested our algorithm on the pre- and post-treatment images for 9 patients where all the slices (approximately 175) showed the presence of GBM.

7.6.2 Benchmark Algorithm - Active Contour Method

We compare the segmentation results of our proposed algorithms to the widely used Chan-Vese Active Contour Method (ACM) [189]. The main goal of this region based method is to minimize the energy function defined by the means of the pixel intensities inside and outside the initial level set curve. Note that this algorithm is not completely automated. The initial level set formulation is conveyed to the algorithm by enabling the user to draw a binary mask over the region of interest in the image. The binary mask is converted to a Signed Distance Function (SDF), such that the region within the curve is assigned positive values, increasing with distance, and the region outside the curve is given increasing negative values, with the distance from the curve. The SDF enables interaction with the energy function as it associates the modification and movement of the initial level set formulation with the change in energy statistics in the two regions. An update occurs with every iteration, wherein the curve evolves and a new SDF is generated based on the previous iteration. The algorithm stops updating the initial level set formulation when the energy is minimized, and further evolution of the curve leads to an increase in the energy value achieved in the previous iteration. Since this algorithm is not based on gradient methods, and deals with balancing the energy on both sides of the curve, it achieves good results even when the image is

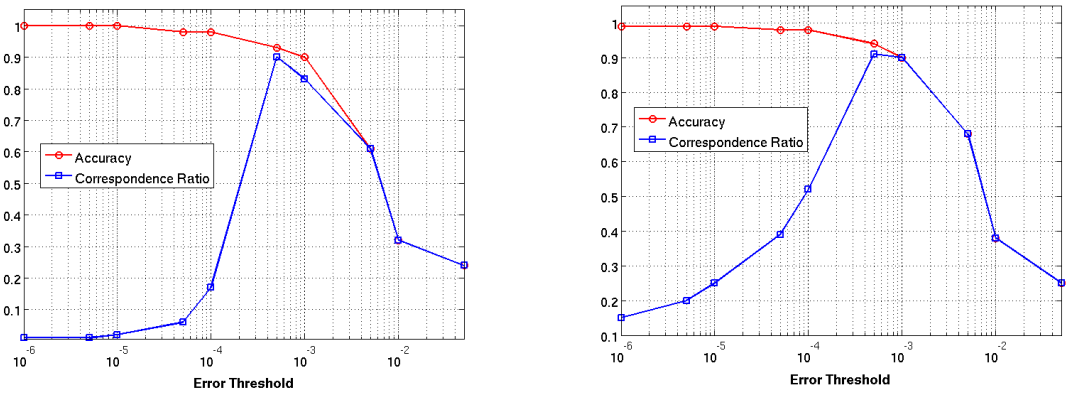


Figure 7.5: Choosing the threshold ϵ for the KSCSA segmentation algorithm. The *Accuracy* and *Correspondence Ratio* are plotted against different values of the error threshold ϵ for two example images. An appropriate threshold, that results in high *Acc* and *CR*, can be chosen using a validation dataset.

blurred. One of the main advantages of this algorithm is that it relies on global properties rather than just taking into account local properties, such as gradients. Furthermore, it provides improved robustness in the presence of noise.

7.6.3 Results

Simulations were carried out independently for both the semi-automated and automated algorithms for every axial slice. For both of the proposed algorithms, the parameter γ for the RBF kernel was set to 0.3, and the dictionary size was fixed at 256. In the automated approach, we computed the ensemble kernel for $15,000$ randomly chosen pixels from the training set. In the reduced complexity semi-automated case, the tumor and non-tumor dictionaries were learned using $10,000$ randomly chosen pixels from tumor and non-tumor regions respectively. The parameter $\beta = 0.1$ was used for sparse coding using the feature sign search algorithm.

The resulting segmented images were compared to the ground truth and performance was measured using the metrics *Accuracy (Acc)* and *Correspondence Ratio (CR)* computed as [188]

$$Acc = \frac{\text{TP}}{\text{Total \# tumor pixels in the GT image}}, \tag{7.14}$$

and

$$CR = \frac{\text{TP} - 0.5\text{FP}}{\text{Total \# tumor pixels in the GT image}}, \tag{7.15}$$

where TP indicates the number of true positives (the pixels indicated as tumorous by the ground truth and our algorithm), and FP denotes the number of false positives (pixels indicated as non-tumorous by the ground truth, but tumorous by our algorithm). The other unknown parameter in the KSCSA approach is the error threshold ϵ, used for classifying the pixels. Figure 7.5 shows the relationship between Acc and CR vs the error threshold (ϵ) for two example images. The ϵ value was fixed at an appropriate value that resulted in high Acc and CR values on a validation dataset.

Figure 7.6 shows the original and segmented images for a few example cases. In each case, the expert-marked ground truth is shown along with the results obtained using the ACM and the proposed algorithms. Both the proposed semi-automated and automated segmentation methods outperformed the benchmark method, and obtained high Acc and CR values as demonstrated by the extensive results in Table 7.1. We observed that the performance of the automated algorithm (KSCA) is equivalent to that of the semi-automated algorithm (KSCSA) in many cases and very closely comparable in the remaining cases. As expected, the semi-automated algorithm is significantly faster when compared to the automated approach. On an average, the proposed semi-automated algorithm takes about 8 seconds (measured using MATLAB R2010b on a 2.8GHz, Intel i7 desktop) in comparison to 120 seconds taken by the automated algorithm. Note that, the average time reported for the semi-automated algorithm does not include the time taken by the user to initialize the tumor region.

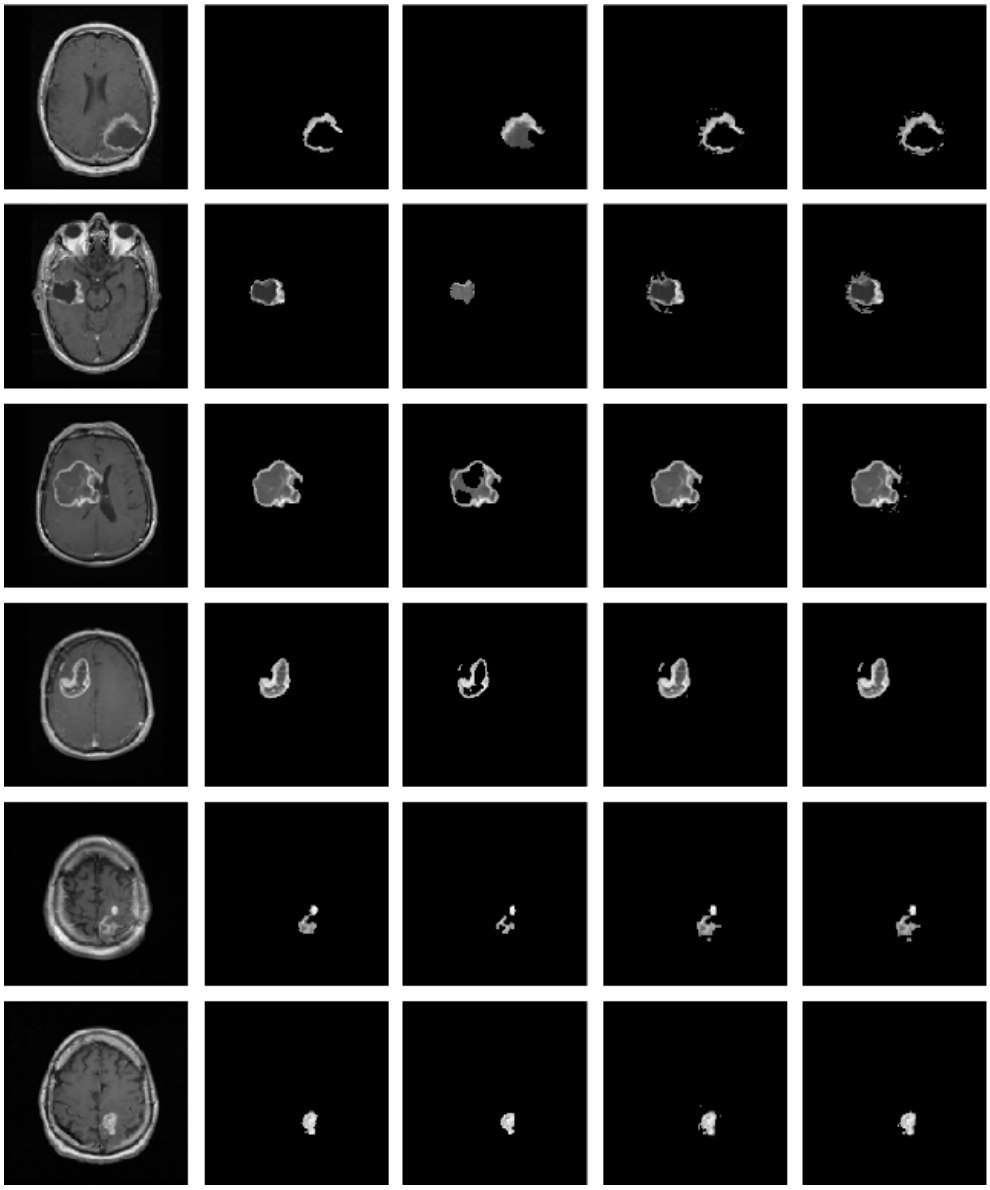


Figure 7.6: Tumor segmentation results. (Left-Right) Original image, Ground Truth (GT) marked by an expert radiologist, Segmentation obtained using the active contour method, Segmentation obtained using the KSCA algorithm, and Segmentation obtained using the KSCSA algorithm. In all cases, the proposed algorithms provide superior quality segmentation when compared to the benchmark algorithm.

Table 7.1: Comparison of the tumor segmentation performance obtained using (a) Active contour method (ACM), (b) Kernel sparse coding-based automated segmentation algorithm (KSCA), and (c) Kernel sparse coding-based semi-automated segmentation algorithm (KSCSA). For each patient, results for a few sample images (pre- and post-treatment) are shown. In each case, the accuracy and correspondence ratio of the segmentation in comparison to expert-marked ground truth are presented.

Image	**ACM**		**KSCA**		**KSCSA**		**Image**	**ACM**		**KSCA**		**KSCSA**	
Set	**Acc**	**CR**	**Acc**	**CR**	**Acc**	**CR**	**Set**	**Acc**	**CR**	**Acc**	**CR**	**Acc**	**CR**
Patient 1:							**Patient 6:**						
Pre	0.81	0.71	0.87	0.86	0.92	0.91	Pre	0.98	0.97	1	0.96	0.99	0.99
Pre	0.42	0.12	0.66	0.33	0.69	0.41	Pre	0.62	0.43	0.96	0.94	0.95	0.94
Pre	0.48	0.22	0.78	0.57	0.78	0.62	Pre	0.87	0.81	0.92	0.91	0.97	0.96
Pre	0.43	0.15	0.72	0.6	0.71	0.64	Post	0.91	0.87	0.92	0.87	0.93	0.91
Pre	0.42	0.13	0.67	0.48	0.68	0.47	Post	0.93	0.89	0.95	0.88	0.95	0.91
Patient 2:							**Patient 7:**						
Pre	0.22	0.16	0.46	0.40	0.49	0.43	Pre	0.44	0.16	0.7	0.62	0.71	0.66
Pre	0.95	0.93	0.96	0.92	0.97	0.93	Pre	0.61	0.41	0.90	0.73	0.90	0.82
Pre	1.00	0.99	1	0.98	0.99	0.99	Pre	0.82	0.73	0.91	0.86	0.90	0.88
Pre	0.87	0.80	0.95	0.81	0.97	0.82	Pre	0.83	0.74	0.90	0.81	0.90	0.79
Pre	0.95	0.93	0.97	0.94	0.98	0.91	Pre	0.94	0.91	0.94	0.92	0.95	0.91
Patient 3:							**Patient 8:**						
Pre	0.97	0.96	0.97	0.96	0.98	0.96	Pre	0.77	0.65	0.95	0.79	0.98	0.87
Pre	0.91	0.86	0.95	0.9	0.98	0.96	Pre	0.73	0.60	0.91	0.8	0.95	0.84
Post	1.00	1.00	0.99	0.97	1	0.99	Post	0.53	0.29	0.92	0.79	0.87	0.82
Post	0.76	0.64	0.98	0.81	0.97	0.85	Post	0.97	0.95	0.97	0.95	0.97	0.95
Post	0.81	0.71	0.83	0.73	0.86	0.72	Post	0.99	0.99	0.99	0.99	0.99	0.99
Patient 4:							**Patient 9:**						
Pre	0.50	0.25	0.64	0.57	0.7	0.65	Pre	0.94	0.91	0.95	0.93	0.95	0.94
Pre	0.53	0.29	0.98	0.84	0.97	0.88	Pre	0.95	0.93	0.98	0.96	0.99	0.94
Pre	0.93	0.90	0.91	0.88	0.92	0.9	Post	0.47	0.21	0.87	0.75	0.88	0.78
Pre	0.40	0.10	0.91	0.82	0.94	0.9	Post	0.63	0.44	0.85	0.84	0.87	0.82
Post	0.73	0.60	0.79	0.67	0.82	0.72	Post	0.82	0.72	0.91	0.88	0.94	0.86
Patient 5:													
Pre	0.94	0.90	0.96	0.88	0.97	0.89							
Pre	0.81	0.71	0.91	0.84	0.90	0.83							
Pre	0.54	0.31	0.68	0.59	0.70	0.66							
Pre	0.92	0.88	0.98	0.96	0.98	0.97							
Pre	0.78	0.66	0.94	0.9	0.95	0.91							

Chapter 8

MEASURING GLOMERULAR COUNT FROM KIDNEY MR IMAGES

8.1 Problem Statement

The variations in the number and size of glomeruli have been linked to several renal and systemic diseases [200, 201]. Though approaches such as acid maceration [202] and the dissector/fractionator stereology technique [203] are commonly used to measure the glomeruli number and size, they require the destruction of the entire kidney. On the other hand, conventional histological methods determine the overall glomeruli statistics by extrapolating the measurements obtained from a few isolated sections. As a result, these methods do not perform direct measurements and cannot localize the identified glomeruli to specific parts of the kidney. Hence, the authors in [204] proposed a robust technique based on magnetic resonance imaging (MRI) to non-destructively measure the glomeruli number and size. This method accurately identifies the glomerulus by injecting cationic ferritin (CF), which causes a decrease in the MRI signal at the location of the glomerulus. The authors demonstrated that the glomerular counts obtained from the 3D MRI images were consistent with the standard histological procedures, while making the measurements in the entire kidney. A sample axial kidney image from a 3-dimensional (3D) magnetic resonance imaging (MRI) data obtained from a cationic ferritin (CF)-injected rat is shown in Figure 8.1.

The goal of this work is to develop an automated algorithm for estimating the glomerular count from a kidney MRI image. A typical solution to solving this problem is to apply a 3D segmentation algorithm over the slices and identify isolated glomeruli, and subsequently count the number of segmented regions. However, glomerular count can also be obtained by considering each axial slice separately. Though the general problem of image segmentation can be unsupervised, we are interested to incorporate some prior knowledge about the MRI image intensities at glomeruli locations. This prior knowledge can be obtained by manually marking glomeruli regions in a few

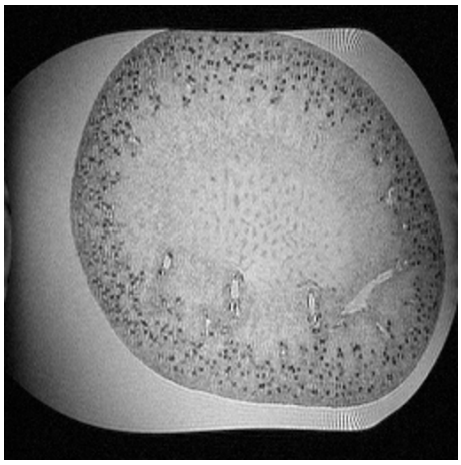


Figure 8.1: An example axial slice from the 3D MRI image obtained from a CF injected rat. The MRI signal is comparatively weak at the locations of the glomerlus.

ground truth images. The other important challenge with unsupervised segmentation algorithms is its computational complexity. By learning a suitable discriminative model from labeled training data, the complexity of segmenting a test image can be reduced significantly, in addition to producing accurate results. Though a variety of segmentation approaches exist in the literature, we propose a novel method based on sparse representations. The proposed approach works by extracting small image patches (size 5×5), and computing a low-dimensional embedding for the patches, such that glomeruli regions are discriminated from the other regions in the MRI image. For a test image, the extracted patches are projected onto the discriminant directions and passed to a clustering algorithm to identify the glomeruli. Extending this approach to the 3D case is straightforward. Instead of extracting patches from a 2D image, we will obtain voxels of size $5 \times 5 \times 5$ by considering 5 consecutive axial slices for each 5×5 patch.

8.2 Background

Mapping data from high-dimensional input spaces to low-dimensional spaces is imperative in several computer vision problems. Several dimensionality reduction procedures work by learning appropriate subspaces to efficiently represent the data. How-

ever, the traditional subspace methods are insufficient when Euclidean distance is incapable of describing the intrinsic similarities between the data samples. Furthermore, the inherent geometry of data in several applications can be non-linear [205, 206]. For example, a set of images that vary in rotation or scale reside on a manifold in the original space. When the data samples are densely distributed on a manifold, we can employ manifold learning methods to infer the underlying inherent structure. A variety of methods that preserve either the global or local properties of the training data have been proposed, and successfully used for dimensionality reduction and visualization. However, mapping a novel test sample to the low-dimensional space is more difficult, when compared to subspace methods. As a result, manifold learning methods are not commonly employed to learn embeddings that can discriminate different classes of data. Some algorithms address this challenge by finding a mapping for the whole data space, not just for the training samples [207–210]. However, these methods preserve data localities or similarities in the low-dimensional embedding space, and hence does not result in class discrimination. As an alternative, the algorithm in [211] considers the geodesic distances between the data samples to perform linear discriminant analysis (LDA).

Another popular approach for describing the relation between training samples is to construct graphs. Several supervised, semi-supervised, and unsupervised machine learning schemes can be unified under the general framework of graph embedding. In addition, subspace approaches such as principal component analysis, linear discriminant analysis, and locality preserving projections can all be posed as graph embedding problems. Two approaches are commonly for graph construction: (i) nearest-neighbor method, where, for each data sample, k nearest neighbors are chosen, and (ii) ϵ-ball based method, where, for each data sample, the samples lying in the ϵ ball surrounding it are chosen. In both cases, the graph edge weights can be fixed as binary values, Gaussian kernel values or the Euclidean distances directly.

Graph embedding can also be applied to supervised and semi-supervised learning problems. Local discriminant embedding (LDE) [212] is a supervised embedding scheme that incorporates intra- and inter- class relationships by respectively defining two different graphs. Furthermore, when only a subset of the training data is labeled, a semi-supervised graph embedding approach, referred to as semi-supervised discriminant analysis (SDA) has been developed [213].

Though the classical graph construction methods have been successful, their performance can be affected by the following reasons: (i) lack of robustness to noise in the data samples. Since the neighbors are identified based on Euclidean distance, noise in even a few data samples can change the graph structure significantly, (ii) inability to identify distinct neighborhood structure for each data sample, when training data is not equally well samples in different areas of the feature space. Since both the graph construction approaches use a fixed global parameter (k or ϵ) to determine the graph structure, they do not allow the use of data-adaptive neighborhoods.

To address the aforementioned challenges, the authors in [34, 163] proposed to use graphs constructed based on sparse codes of the data samples. Sparse coding aims to obtain a parsimonious representation for the data using the basis functions in a given dictionary. When constructing the ℓ_1 graph, we use the set of training samples as the dictionary directly, instead of inferring from the data. It has been demonstrated that sparse coding is robust to noise, and does not include unrelated inhomogeneous data since the choice of neighbors is data dependent. The sparse codes are obtained using ℓ_1 minimization and hence the complexity of constructing the graph is quite high when compared to the classical graph construction approaches. Furthermore, the ℓ_1 graph construction approach is unsupervised and does not take into account the label information of the training data.

In this paper, we propose to obtain discriminative embeddings for patches from the kidney MRI images, using ℓ_1 graphs. Though our formulation is for a

2−class case, this can be generalized to multi-class problems as well. The proposed method allows us to incorporate prior knowledge from the expert-marked ground truth images and eliminates the need to construct the ℓ_1 graphs for the test images. As a result, the process of identifying the glomeruli regions, and subsequently obtaining the count, is computationally efficient and provides accurate results in comparison to other segmentation approaches.

8.3 Supervised Graph Embedding

The inherent low dimensionality of data is usually exploited in order to achieve improved performances in various computer vision and pattern recognition tasks. In the recent years, the use of non linear dimensionality reduction (NLDR) techniques has gained lot of interest. Their main aim is to identify a low dimensional manifold onto which high dimensional data can be projected while preserving most of the local and/or global structure. A class of approaches known as manifold clustering and manifold learning algorithms such as Locally Linear Embedding [214], Hessian LLE [215] and Laplacian Eigenmaps [216] preserve local information of the manifold. Global methods such as Isomap [217] and semidefinite embedding [218] try to preserve global and local relationships. In addition, approaches such as Locality Preserving Projections (LPP) [219] explicitly learn mapping functions to project data onto low-dimensional spaces. Although the learned embedding is applicable to novel test samples, these algorithms are more appropriate for clustering problems wherein no supervised information is available. Interestingly, several of these embedding algorithms can be unified under the framework of graph embedding [220].

LPP is an unsupervised graph embedding approach that computes projection directions, such that the pairwise distances of the projected training samples in the neighborhood are preserved. Let us define the training data as $\{\mathbf{x}_i|\mathbf{x}_i \in \mathbb{R}^M\}_{i=1}^T$. An undirected graph G is defined, with the training samples as vertices, and the similarity

between the neighboring training samples coded in the affinity matrix $\mathbf{W} \in \mathbb{R}^{T \times T}$. Let us denote the graph Laplacian as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is a degree matrix with each diagonal element containing the sum of the corresponding row or column of $\mathbf{L}$. The d projection directions for LPP, $\mathbf{V} \in \mathbb{R}^{M \times d}$, can be computed by optimizing

$$\min_{\text{trace}(\mathbf{V}^T\mathbf{X}\mathbf{D}\mathbf{X}^T\mathbf{V})=\mathbf{I}} \text{trace}(\mathbf{V}^T\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{V}). \tag{8.1}$$

Here $\mathbf{X}$ is a matrix obtained by stacking all data samples as its columns. The embedding for any data sample $\mathbf{x}$ can be obtained by projecting onto the orthonormal directions $\mathbf{V}$ as $\mathbf{V}^T\mathbf{x}$. It can be shown that LPP computes the projection directions by minimizing the objective

$$\min_{\mathbf{V}} \sum_{i,j=1}^{T} \|\mathbf{V}^T\mathbf{x}_i - \mathbf{V}^T\mathbf{x}_j\|_2^2 w_{ij} \text{ s.t. } \sum_{i=1}^{T} \|\mathbf{V}^T\mathbf{x}_i\|_2^2 \delta_{ij} = 1. \tag{8.2}$$

This optimization ensures that the embedding preserves the neighborhood structure of the graph.

However, there is a need to incorporate class label information to learn an embedding that can discriminate different classes of data. Linear Discriminant Analysis (LDA) is such a supervised graph embedding framework that uses the within-class and between-class scatter matrices to obtain discrimiantive projections. Let us assume that the class labels corresponding to the set of training samples, $\{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{R}^M\}_{i=1}^T$, are known and denoted by $\{y_i | y_i \in \{1, 2, \ldots, C\}\}_{i=1}^T$. Here C indicates the total number of classes. In order to pose LDA as a graph embedding problem, we compute the intra-class and inter-class graphs as follows.

$$w_{ij} = \begin{cases} \frac{1}{T_{y_i}} & \text{if } y_i = y_j, \\ 0 & \text{otherwise.} \end{cases} \tag{8.3}$$

Here T_{y_i} denotes the total number of samples with the label y_i.

$$w'_{ij} = \frac{1}{T}. \tag{8.4}$$

Using these affinity matrices, we construct the diagonal degree matrices $\mathbf{D}$ and $\mathbf{D}'$, whose elements are computed as $d_{ii} = \sum_j w_{ij}$ and $d'_{ii} = \sum_j w'_{ij}$ respectively. Finally, the graph Laplacian matrices are obtained as $\mathbf{L} = \mathbf{D} - \mathbf{W}$ and $\mathbf{L}' = \mathbf{D}' - \mathbf{W}'$. Given the two graph laplacian matrices, the low-dimensional embedding is computed as

$$\max_{\mathbf{V}} \frac{\text{Tr}[\mathbf{V}^T\mathbf{L}'\mathbf{V}]}{\text{Tr}[\mathbf{V}^T\mathbf{L}\mathbf{V}]}. \tag{8.5}$$

Note that, the reduced number of dimensions d is fixed at $C-1$. The above optimization computes an embedding such that the between class scatter is maximized, while minimizing the within-class scatter. LDA aims to solve a trace-ratio maximization problem and this can be converted to an equivalent ratio-trace maximization problem, $\text{Tr}[(\mathbf{V}^T\mathbf{L}\mathbf{V})^{-1}\mathbf{V}^T\mathbf{L}'\mathbf{V}]$. A greedy solution for this problem can be obtained using the generalized eigen value decomposition

$$\mathbf{X}\mathbf{L}'\mathbf{X}^T = \lambda\mathbf{X}\mathbf{L}\mathbf{X}^T. \tag{8.6}$$

The set of directions $\mathbf{V}$ is computed as the d top eigen vectors.

Alternatively, the affinity matrices can be constructed by considering both the class label information and the local structure of the data samples. Let $\mathcal{N}_k(i)$ denote the set of $k-$nearest neighbors for the data sample $\mathbf{x}_i$. The affinity matrices are then constructed as

$$w_{ij} = \begin{cases} 1 & \text{if } y_i = y_j \text{ AND } [i \in \mathcal{N}_k(j) \text{ OR } j \in \mathcal{N}_k(i)], \\ 0 & \text{otherwise.} \end{cases} \tag{8.7}$$

$$w'_{ij} = \begin{cases} 1 & \text{if } y_i \neq y_j \text{ AND } [i \in \mathcal{N}'_k(j) \text{ OR } j \in \mathcal{N}'_k(i)], \\ 0 & \text{otherwise.} \end{cases} \tag{8.8}$$

Given the affinity matrices, the embedding can be obtained as in the case of LDA. This approach is referred to as local discriminant embedding (LDE).

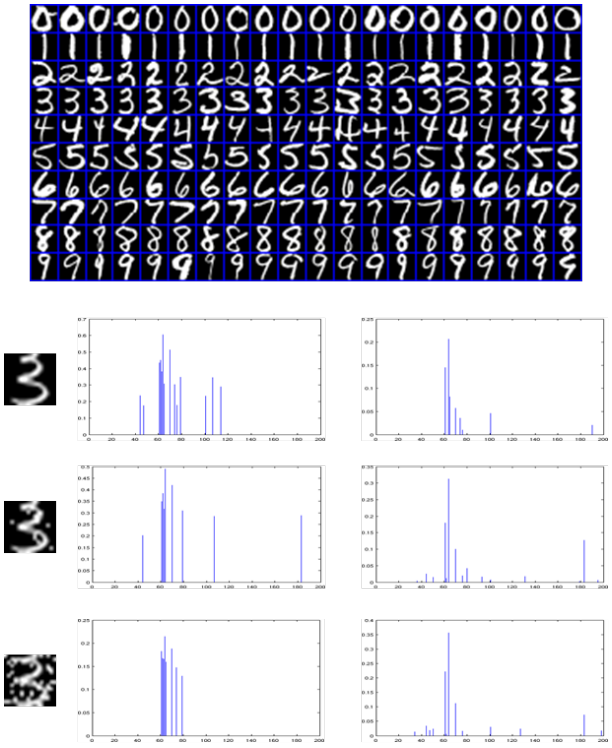


Figure 8.2: Demonstration of the robustness of an ℓ_1 graph. The set of training samples used for this simulation are obtained from the USPS dataset [5]. For an example data sample (Digit 3), its similarities to all data samples in the case of a K-Nearest Neighbor graph (left) and an ℓ_1 graph (right) are shown. As the data sample is corrupted by noise, the K-NN graph changes significantly while the ℓ_1 is robust to the noise.

8.4 Proposed Algorithm

In this section, the proposed algorithm for measuring the glomerular number is described. By using sparse codes to define the relation between data samples, more robust graphs can be constructed for unsupervised and supervised learning tasks. In particular, we propose to perform discriminative embedding using ℓ_1 graphs. We begin by discussing the construction of ℓ_1 graphs and subsequently the training and testing stages of the proposed algorithm.

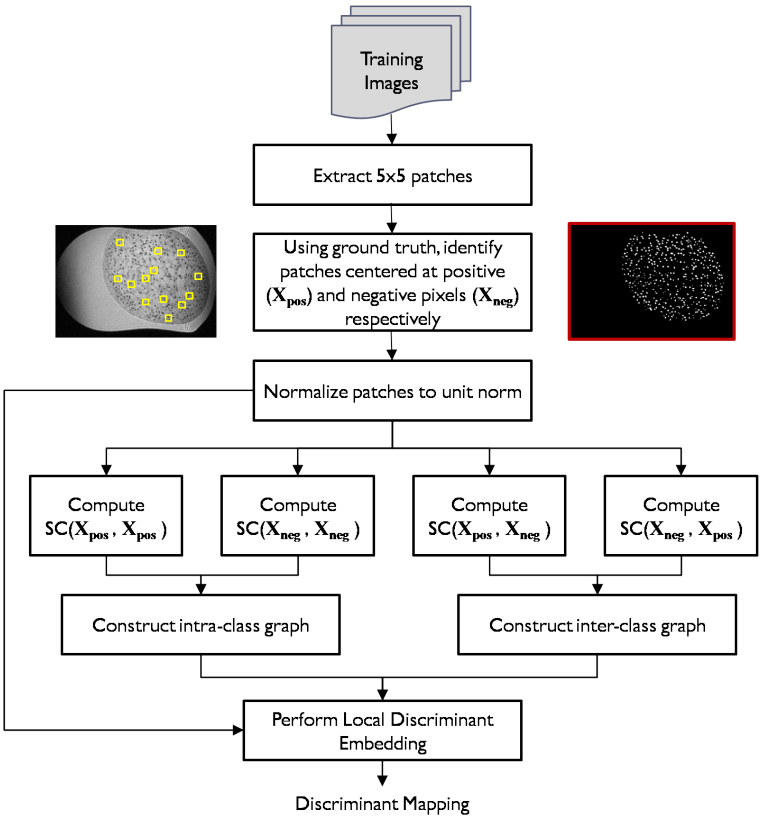


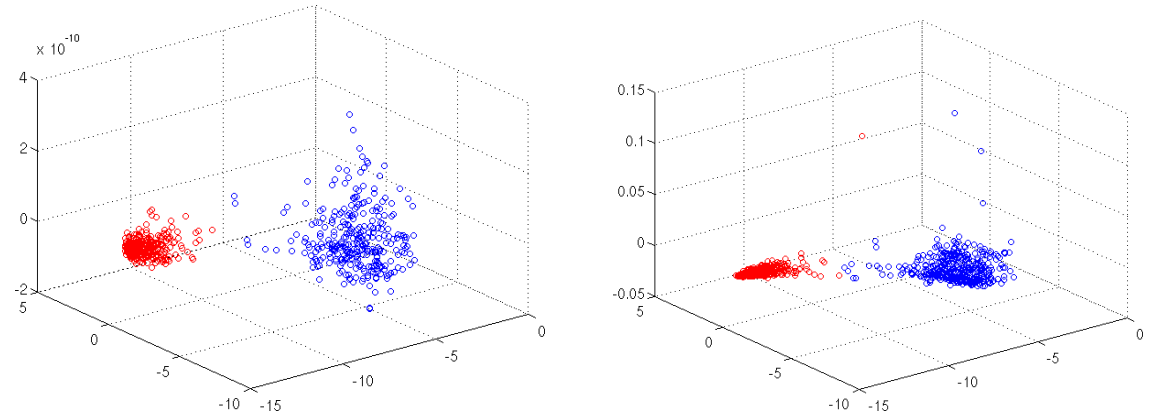
Figure 8.3: Proposed algorithm for computing an embedding that discriminates image patches containing glomeruli from the rest. This method works by constructing ℓ_1 graphs to model inter-class and intra-class relationships, and performing local discriminant embedding.

8.4.1 Constructing ℓ_1 Graphs

In sparse modeling, data is represented as a sparse linear combination of atoms from a "dictionary" matrix. The dictionary is typically overcomplete, i.e., the number of basis functions exceeds the data dimension and hence we need to solve an underdetermined system of linear equations. The linear generative model for sparse modeling of a data sample $\mathbf{x} \in \mathbb{R}^M$ is given by,

$$\mathbf{x} = \mathbf{\Psi}\mathbf{a} \tag{8.9}$$

where $\mathbf{\Psi} \in \mathbb{R}^{M \times K}$ is the set of K elementary features and $\mathbf{a} \in \mathbb{R}^K$ is the coefficient vector. Though different forms of regularization can be applied to this problem, a



(a) Training Images (b) Test Images

Figure 8.4: An example demonstration for the proposed supervised graph embedding approach. This simulation uses 2 classes of digits from the USPS dataset, and divides them into train/test sets. Using the training images, we compute the discriminant mapping and obtain the low-dimensional features for both the training and test images. The compactness of the classes observed in the test images reflects the discrimination power of the proposed embedding.

sparse solution is more robust and can be effective for recovering the data sample $\mathbf{x}$. If we assume that the coefficient vector is sparse and has statistically independent components, the elements of the dictionary $\mathbf{\Psi}$ can be inferred from the generative model using appropriate constraints. Considering the generative model for sparse coding (8.9), the codes can be obtained either by minimizing the exact ℓ_0 penalty or its convex surrogate ℓ_1 penalty as,

$$\text{(PL0)}\ \hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmin}} \|\mathbf{a}\|_0 \text{ subj. to } \mathbf{x} = \mathbf{\Psi}\mathbf{a}, \tag{8.10}$$

$$\text{(PL1)}\ \hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmin}} \|\mathbf{a}\|_1 \text{ subj. to } \mathbf{x} = \mathbf{\Psi}\mathbf{a}, \tag{8.11}$$

where $\|.\|_0$ is the ℓ_0 norm and $\|.\|_1$ is the ℓ_1 norm. Since real-world data cannot be expressed exactly using the generative model in (8.9), usually the equality constraints in (8.10) and (8.11) are replaced using the constraint $\|\mathbf{x} - \mathbf{\Psi}\mathbf{a}\|_2^2 \leq \epsilon$, where ϵ is the error goal of the representation. The exact ℓ_0 minimization given in (8.10) is a combinatorial problem and that is the major reason why its convex surrogate is often used.

Given a set of unlabeled training samples $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{T}$, the ℓ_1 graph can be constructed as follows. For each data sample $\mathbf{x}_i$, we solve the ℓ_1 optimization problem to obtain the sparse codes,

$$\min_{\mathbf{a}_i} \|\mathbf{a}_i\|_1, \text{ s.t. } \mathbf{x}_i = \mathbf{\Psi}^i \mathbf{a}_i. \tag{8.12}$$

Here the dictionary $\mathbf{\Psi}^i = [\mathbf{x}_1, \ldots, \mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_T]$ is designed as the set of training samples, leaving out the sample $\mathbf{x}_i$. The importance of a training sample in representing another sample is used as the indicator for similarity between these two samples. Hence, representation coefficients can be used to build the similarity matrix for the graph. Since the entries of this weight matrix represent the similarities between data samples, we can assume them to be non-negative. Hence, the ℓ_1 minimization can be solved with non-negative constraints on the coefficient values. Let us denote this process of computing the sparse coefficient matrix as $\mathbf{A} = \text{SC}(\mathbf{X}, \mathbf{X})$, where the first argument is the data matrix, and the second argument is the dictionary matrix. Using the sparse coefficient matrix, $\mathbf{A} \in \mathbb{R}^{T \times T}$, as the similarity matrix, we construct the laplacian for the graph as $\mathbf{L} = (\mathbb{I} - \mathbf{A})^T(\mathbb{I} - \mathbf{A})$. Here, $\mathbb{I}$ denotes the identity matrix of size $T \times T$. This laplacian matrix can be subsequently used to perform spectral clustering. In order to demonstrate the robustness of sparse representations, we consider a subset of images from the USPS handwritten digit dataset. Figure 8.2 shows the similarities between one of the training samples and the rest, obtained using nearest neighbors and sparse representations. Clearly, using sparse coding provides a more robust graph even when the data sample is noisy. In addition to building ℓ_1 graphs, graph embedded sparse codes can be obtained as described in [161].

8.4.2 Incorporating Supervisory Information

The ℓ_1 graph construction is unsupervised, and hence supervisory information cannot be incorporated. The problem of identifying glomeruli in the kidney images can be solved using spectral clustering with ℓ_1 graphs. However, such an approach poses two

main challenges: (i) prior knowledge about the intensities in the glomeruli regions cannot be used, and (ii) computational complexity of building an ℓ_1 graph is high. In this paper, we propose to obtain an embedding using a set of training images and build a low-complexity method for estimating the glomerular number in any test image.

In order incorporate supervisory information, we first build a training set by manually marking the glomerular regions in a few kidney MRI images. The training stage of the algorithm works with image patches of size 5×5 extracted from the training images. Patches are extracted around every pixel in the image, and those patches centered at the pixels marked as belonging to glomeruli regions are considered to be positive examples ($\mathbf{X}_{pos}$). The rest of the patches are marked as negative examples ($\mathbf{X}_{neg}$). Our goal is to create a low-dimensional embedding that discriminates the positive and negative examples effectively. Local discriminant embedding addresses this problem by building graphs as described in Section 8.3. In order to obtain a more robust embedding, we propose to use ℓ_1 graphs.

As a preprocessing step, all training patches are normalized to unit ℓ_2 norm. Optionally, mean removal can also be performed prior to normalization. Following this, we compute four different similarity (sparse coefficient) matrices for constructing the ℓ_1 graph using the procedure described in Section 8.4.1: (i) $\mathbf{A}_{p,p} = \mathrm{SC}(\mathbf{X}_{pos}, \mathbf{X}_{pos})$, (ii) $\mathbf{A}_{p,n} = \mathrm{SC}(\mathbf{X}_{pos}, \mathbf{X}_{neg})$, (iii) $\mathbf{A}_{n,p} = \mathrm{SC}(\mathbf{X}_{neg}, \mathbf{X}_{pos})$, and (iv) $\mathbf{A}_{n,n} = \mathrm{SC}(\mathbf{X}_{neg}, \mathbf{X}_{neg})$. In order to use these similarities to discriminate between the two classes, we propose to construct intra-class and inter-class similarity matrices as follows.

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{p,p} & \mathbf{0}_{T_p,T_n} \\ \mathbf{0}_{T_n,T_p} & \mathbf{A}_{n,n} \end{bmatrix} \tag{8.13}$$

$$\mathbf{A}' = \begin{bmatrix} \mathbf{0}_{T_p,T_p} & \mathbf{A}_{p,n} \\ \mathbf{A}_{n,p} & \mathbf{0}_{T_n,T_n} \end{bmatrix} \tag{8.14}$$

The corresponding laplacian matrices are computed as

$$\mathbf{L} = (\mathbb{I} - \mathbf{A})^T(\mathbb{I} - \mathbf{A}), \tag{8.15}$$

$$\mathbf{L}' = (\mathbb{I} - \mathbf{A}')^T(\mathbb{I} - \mathbf{A}'). \tag{8.16}$$

By creating a low-dimensional projection $\mathbf{V}$, we are interested in preserving the intra-class relationships, while making samples from different classes to be dissimilar. This is achieved by performing local discriminant embedding with $\mathbf{L}$ and $\mathbf{L}'$ respectively as shown in (8.5). Figure 8.3 describes the process of computing a discriminant embedding using ℓ_1 graphs. In order to demonstrate the behavior of the proposed embedding approach, we build an example dataset using images from 2 different classes (Digits 3 and 7) in the USPS dataset. In each class, we use 300 randomly chosen images for training and the rest for testing. Figure 8.4(a) shows the 3−dimensional embedding of the training images and it can be seen that the two classes are well separated. By using the identified mapping for novel test images, we obtain the features in Figure 8.4(b). The compactness of samples within a class, and separation between the two classes reflects the discrimination power of the proposed embedding.

8.4.3 Obtaining Glomerular Count

Since we have learned a discriminant embedding, computing low-dimensional features for a test image is straightforward. Unlike conventional sparse coding methods, we need not compute sparse codes for the test image patches. As a result, the proposed algorithm works in near real-time, and processes an image of 256×256 in just around 0.2 seconds. Given a test image, we extract 5×5 patches similar to the training stage, and normalize them to unit ℓ_2 norm. Following this we project the vectorized patches onto the discriminant directions as $\mathbf{V}^T\mathbf{Z}$, where $\mathbf{Z}$ denotes the matrix of patches from

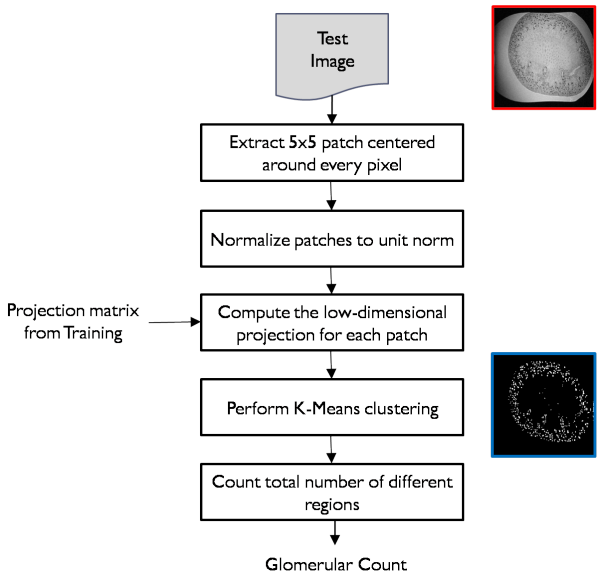


Figure 8.5: A low-complexity procedure for obtaining the glomerular count in a test image. The discriminant mapping determined in the training stage is used with the test image patches directly, and hence there is no need to obtain the sparse codes.

a test image. Using these low-dimensional features, we employ the simple K-means clustering procedure to perform segmentation. Given the segmented image, we count the number of independent regions with atleast more than r pixels. This implies that we assume that the glomeruli regions contain at least r pixels in its neighborhood, so that we can ignore a few lonely false positives provided by the algorithm. The steps involved in obtaining the glomerular count for a test image is illustrated in Figure 8.5.

8.5 Experimental Results

8.5.1 Dataset

For evaluating the performance of the proposed algorithm, we use the dataset reported by the authors in [204]. Furthermore, the estimated glomerular number is compared against the results obtained using the MRI segmentation used in [204], acid maceration, and stereological methods. We now briefly describe the procedure for obtaining the dataset. Additional details on the materials and methods can be

found in [204]. Cationic Ferritin was synthesized and male Sprague-Dawley rats, weighing between 215 and 245 g, were given three intravenous bolus doses. Kidneys were perfused and fixed via transcardial perfusion of PBS followed by 10% neutral buffered formalin, then resected and stored in glutaraldehyde. The perfused left kidneys were imaged in glutaraldehyde on a Varian 19T 89-mm-bore NMR (Varian, Palo Alto, CA), equipped with a DOTY 3-axis imaging probe and a gradient with a maximum strength of 300 G/cm (DOTY Scientific, Columbia, SC). Scans were acquired with a 3D gradient echo (GRE) sequence with echo time/repetition time = 7/40 ms and a resolution of $62 \times 62 \times 78\mu$m. Total scan time was 6 hours/kidney.

8.5.2 Benchmark Method

Labeled glomeruli in the 3D MRI data set can be counted using the method used in [204]. Using bicubic interpolation, images in the data set were resized and the spatial resolution was changed to $31 \times 31 \times 62\mu$m. Spatial signal magnitude gradients were computed to identify significant spatial changes in signal magnitude throughout the volume. Only voxels exceeding a threshold on the signal magnitude difference were included for the subsequent operations. Following this, regional minima were located in these areas using an upper signal magnitude threshold. Regions in the image considered to be glomeruli were then labeled based on morphological thresholds. Note that, here it is assumed that a glomerulus is approximately spherical. Finally, the watershed transform was computed on these regions to distinguish individual glomeruli where signal overlap of multiple glomeruli might occur.

8.5.3 Results

In this section, we report the results obtained using the proposed algorithm, in comparison to the benchmark technique with MRI images, acid maceration, and stereology methods. The evaluation was carried out with three different subjects (*Rat A*, *Rat B* and *Rat C*) and the glomerular count for the MRI based approaches were

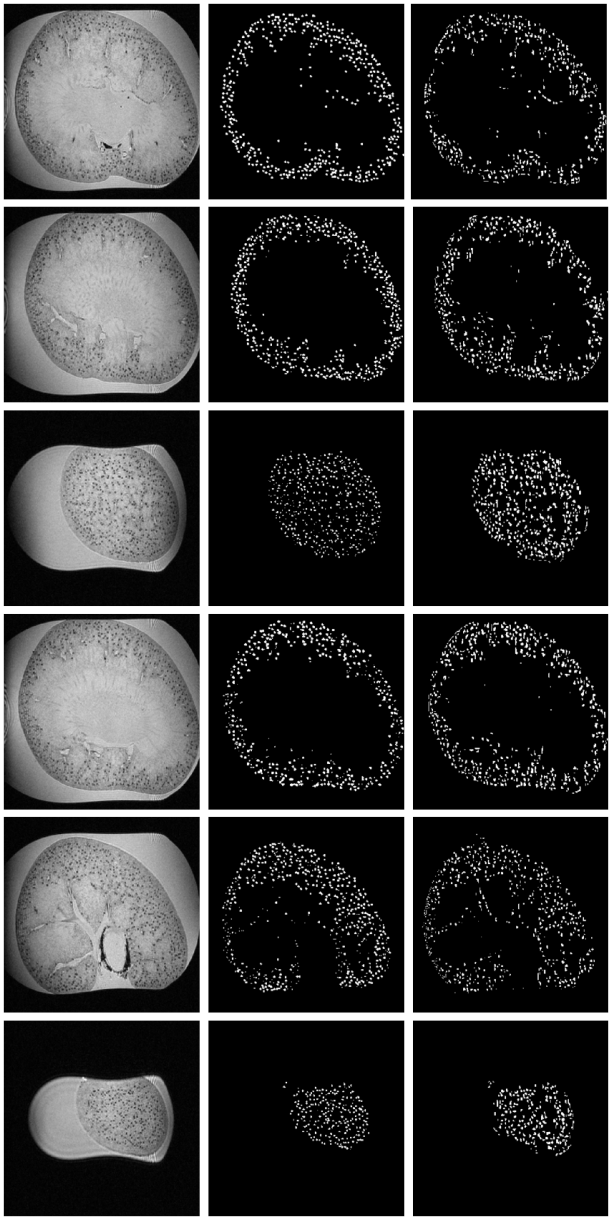


Figure 8.6: Segmentation results for a sample set of MRI images obtained using the proposed algorithm. (Left-Right) Original kidney image with reduced intensities at glomeruli locations; Ground truth image with manually labeled glomeruli regions; Segmentation obtained using the approach described in Section 8.4.3.

obtained from 192 slices. Note that the results reported for the proposed algorithm were obtained using only 2-D slices. For training the discriminant mapping, 5 ground truth images were used and patches of size 5×5 were extracted. Using the ground truth labels, a training set containing $7,500$ positive and $7,500$ negative example

Table 8.1: Glomerular count obtained using acid maceration, stereology, and the MRI techniques, for three different datasets. Each dataset consisted of 192 images and total glomerular count is shown. In addition, the total time taken, in seconds, to process each dataset using the proposed algorithm is shown.

Dataset	**MRI Number [204]**	**Maceration Number**	**Stereology Number**	**Proposed Algorithm**	**Time Taken (s)**
Rat A	$32,789$	$27,504$	$34,504$	$33,884$	41.1
Rat B	$35,203$	$31,190$	$35,421$	$34,335$	42.4
Rat C	$31,772$	$31,075$		$32,117$	44.9
Rat D	$39,482$	$33,321$		$34,765$	40.3
Rat E	$29,682$	$31,478$		$35,208$	46.6
Mean	$33,786$	$30,914$	$34,963$	$34,062$	43.1
STD	$3,753$	$2,112$	648	1193	2.6

patches was generated. The patches were normalized, and the four ℓ_1 graphs were constructed with the sparsity penalty λ fixed at 0.2. The number of reduced dimensions d was fixed at 10 and the mapping $\mathbf{V} \in \mathbb{R}^{25\times 10}$ was obtained using the algorithm described in Figure 8.3. For each image in the evaluation dataset, the normalized 5×5 patches were projected onto the discriminant directions, and K-means clustering was performed to identify the gomeruli. Finally, the number of independent regions was counted and reported as the glomerular number.

Figure 8.6 illustrates the segmentation obtained for a set of test images, using the proposed algorithm. In each case, both the original and ground truth images are shown for comparison. Clearly, the proposed algorithm provides impressive results in determining the glomeruli regions and the glomeruli count is comparable to the manual count. The glomerular counts for the three evaluation datasets are shown in Table 8.1. The results obtained using the proposed method are comparable to the stereology results, and the variance is comparatively lower. Another important advantage of the proposed scheme is its low computational complexity, which is evident from the time reported in 8.1. The time complexity was measured in MATLAB R2010b on a 2.8GHz, Intel i7 desktop.

Chapter 9

LOCAL SPARSE CODING IN OBJECT RECOGNITION AND IMAGE RETRIEVAL

Several state-of-the-art object recognition systems are based on Spatial Pyramid Matching (SPM) [99]. An image is partitioned into increasingly finer regions and features are evaluated in the local regions. Local descriptors are extracted from small image patches, coded with a dictionary learned using features from several training images, and the code vectors in each spatial region are pooled together by building histograms. Finally, the histograms of the different spatial regions are concatenated and presented to a non-linear classifier. Typically, the features extracted from the images are either Scale Invariant Feature Transform (SIFT) descriptors or Histogram of Oriented Gradients (HOG) descriptors. The descriptors are coded using vector quantization with a K-means dictionary. Though, this approach has been very effective, the complexity of using a non-linear classifier is quite high. Hence, the authors in [100] proposed to replace the vector quantization in SPM by sparse coding, which enabled the use of linear classifiers. Furthermore the authors of [167] showed that sparse representations can be efficiently performed in a high dimensional feature space using the kernel trick. The kernel trick maps the non-linear separable features into a high dimensional feature space, in which similar features are grouped together, hence improving the linear separability.

Note that sparse coding algorithms aim to reduce only the reconstruction error and hence do not explicitly consider the correlation between the codes, which is crucial for classification tasks. As a result, the codes obtained for similar features may be quite different. The overcomplete nature of the dictionary makes the sparse coding process highly sensitive to the variance in the features. This can be circumvented by suitably regularizing the sparse coding problem. Laplacian sparse

coding [160] addresses this by exploiting the dependence between the local features, thereby ensuring the consistence of the sparse codes. In [154], the authors proposed the locality-constrained linear coding approach, which explicitly encourages the code to have non-zero coefficients for dictionary atoms in the neighborhood of the encoded data. In [153] it was shown theoretically that, under certain assumptions, locality is more important than sparsity for non-linear function learning using the obtained local codes.

9.1 Locality in Sparse Models

In non-linear function learning problems, it is typical to overfit the data particularly when the dimensionality of the data is much higher in comparison to the number of data samples. However, in several real problems we do not observe this so-called curse of dimensionality, since they often lie on a manifold with much smaller intrinsic dimensionality. Performing machine learning tasks using such data is to effectively exploit the intrinsic geometric properties of the manifold of interest. In recent years, the knowledge of the geometric properties have been utilized to devise sophisticated algorithms for tasks such as classification and clustering. One of the approaches that has been popular in the machine learning community is the use of non linear dimensionality reduction (NLDR) techniques [221]. Their main aim is to identify a low dimensional manifold onto which the high dimensional data can be projected while preserving most of the local and/or global structure. Approaches such as Locally Linear Embedding, Hessian LLE, and Laplacian Eigenmaps preserve local information of the manifold. In contrary, global methods such as Isomap, semidefinite embedding etc. try to preserve both global and local relationships. However, this class of techniques assumes that the data is embedded in a high-dimensional ambient vector space, i.e. the manifold is an embedded sub-manifold of some vector space. If such an embedding can be found, one can apply a variety of methods developed for

vector-spaces in conjunction with one of several NLDR techniques. Locally Linear Embedding (LLE) is an unsupervised learning algorithm which exploits the fact that, the local geometry of a non-linear function can be well approximated using a linear model. When the dictionary $\mathbf{D}$ represents the set of anchor points that characterize the local geometry, the idea of using sparse coding to model the local neighborhood of data merits attention. However, sparse coding, in the absence of additional constraints, tries to reduce the error of the representation without any consideration on locality. It is possible to include additional locality constraints by considering the general class of the weighted ℓ_1 minimization problems,

$$\min_{\mathbf{x}} \sum_{k=1}^{K} w(k)|x_k| \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{\Psi}\mathbf{x}\|_2 \leq \epsilon, \tag{9.1}$$

where $w(1), ..., w(K)$ are positive weights. It can be clearly seen that large weights could be used to encourage zero entries in the sparse code $\mathbf{x}$. The LCC algorithm proposed in [153] computes local sparse codes using weights based on the Euclidean distance measure as given by

$$w(k) = \|\mathbf{y} - \boldsymbol{\psi}_k\|_2^2, \tag{9.2}$$

where $\boldsymbol{\psi}_k$ is the k^{th} column of $\mathbf{\Psi}$. Since the dictionary elements are normalized, the following distance metric can be alternatively employed to compute the neighborhood.

$$w(k) = \|\mathbf{y} - (\mathbf{y}^T\boldsymbol{\psi}_k)\boldsymbol{\psi}_k\|_2^2. \tag{9.3}$$

Note that this metric proposed in [45] is used by K-hyperline clustering [35] to identify the membership of a data sample. In K-hyperline clustering, a cluster center is evaluated as the rank-1 SVD (singular value decomposition) of the data samples belonging to that cluster. This weighting scheme directly considers the coherence between the normalized dictionary elements and the data sample $\mathbf{y}$.

Since the weighted ℓ_1 minimization in (9.1) is computationally expensive, an approximate method for locality constrained linear coding (LLC) was proposed in

[154]. The LLC algorithm employs the following criteria:

$$\min_{\mathbf{X}} \sum_{i=1}^{T} \|\mathbf{y}_i - \mathbf{\Psi}\mathbf{x}_i\|_2^2 + \lambda \|\mathbf{w}_i \odot \mathbf{x}_i\|_2^2 \text{ subject to } \mathbf{1}^T\mathbf{x}_i = 1, \forall i, \tag{9.4}$$

where $\odot$ denotes element-wise multiplication and $\mathbf{w}_i$ measures the similarity between the i^{th} data sample and all the dictionary atoms. The distance metric used is

$$w_i(k) = \exp\left(\frac{\|\mathbf{y}_i - \boldsymbol{\psi}_k\|_2}{\sigma}\right), \forall k, \tag{9.5}$$

where σ is used to adjust the control the magnitude of weights in the neighborhood of a data sample. In order to speed up this procedure, the P nearest dictionary atoms are first identified and a smaller linear system is solved using a least squares procedure on the chosen dictionary atoms. This reduces the computational complexity from $\mathcal{O}(K^2)$ to $\mathcal{O}(K + P^2)$, where K denotes the number of dictionary atoms and $P << K$.

9.2 Local Sparse Coding

In this section, we describe the proposed dictionary learning algorithm for local sparse coding of image descriptors. Similar to the dictionary learning approaches for regular sparse coding, the proposed algorithm iterates between local sparse coding and dictionary update steps. However, we need to take into account the neighborhood relation between a dictionary atom and the training vectors it represents, when updating the dictionary. The dictionary learning is a generalized clustering procedure where the training vectors are assigned to more than one dictionary atom in the local sparse coding step. Similarly, the training vectors participate in the update of more than one dictionary atom in the update step. The dictionary is initialized using cluster centroids obtained from K-means clustering of the training vectors. The joint optimization problem for local sparse coding and dictionary learning can be expressed as

$$\left\{\hat{\mathbf{\Psi}}, \{\hat{\mathbf{x}}_i\}_{i=1}^{T}\right\} = \min_{\mathbf{\Psi}, \{\mathbf{x}_i\}_{i=1}^{T}} \sum_{i=1}^{T} \|\mathbf{y}_i - \mathbf{\Psi}\mathbf{x}_i\|_2^2 + \|\mathbf{W}_i\mathbf{x}_i\|_1, \tag{9.6}$$

additionally constraining $\|\boldsymbol{\Psi}_k\|_2 = 1, \forall k = \{1, \ldots, K\}$. In order to solve for local sparse codes, we rewrite the weighted minimization problem in (9.6) as

$$\hat{\mathbf{x}}_i = \underset{\mathbf{x}_i}{\operatorname{argmin}} \|\mathbf{y}_i - \boldsymbol{\Psi}\mathbf{W}_i^{-1}\mathbf{x}_i\|_2^2 + \lambda\|\mathbf{x}_i\|_1, \forall i = \{1, \ldots, T\}. \tag{9.7}$$

Here $\mathbf{W}_i$ is a diagonal matrix containing the weights corresponding to the dictionary elements in $\boldsymbol{\Psi}$ computed as

$$\mathbf{W}_i(k,k) = \|\mathbf{y}_i - \boldsymbol{\psi}_k\|_2^2, \forall k = \{1, \ldots, K\}. \tag{9.8}$$

Setting $\boldsymbol{\Gamma}_i = \boldsymbol{\Psi}\mathbf{W}_i^{-1}$, (9.7) can be expressed as

$$\hat{\mathbf{x}}_i = \underset{\mathbf{x}_i}{\operatorname{argmin}} \|\mathbf{y}_i - \boldsymbol{\Gamma}_i\mathbf{x}_i\|_2^2 + \lambda\|\mathbf{x}_i\|_1, \forall i = \{1, \ldots, T\}. \tag{9.9}$$

This is equivalent to standard sparse coding with the modified dictionary $\boldsymbol{\Gamma}_i$. We can employ the feature-sign search algorithm or the LARS-LASSO method to efficiently solve (9.9) for the sparse codes. Though these algorithms are significantly faster than using convex optimization, the following greedy approach can also be used to evaluate the local code for test data. For a test data sample $\mathbf{y}$, we can first identify the P nearest neighbors and then compute the corresponding $P-$sparse code using

$$\min_{\mathbf{x}} \left\| \mathbf{y} - \sum_{k \in \mathcal{C}} \boldsymbol{\psi}_k x_k \right\|_2 \quad \text{subject to} \quad \|\mathbf{x}\|_0 \leq P, \tag{9.10}$$

where $\mathcal{C}$ contains the set of indices of the P nearest dictionary atoms selected by the distance metric in (9.3). This optimization implies that only the selected dictionary atoms, $\{\boldsymbol{\psi}_k\}_{k \in \mathcal{C}}$, can participate in the approximation of $\mathbf{y}$. This can be solved using the Orthogonal Matching Pursuit or the LARS algorithm.

9.2.1 Dictionary Learning

As described earlier, updating the dictionary atom should not affect its neighborhood relation with the training vectors it represents when trying to reduce the error. The non-zero coefficients in the local sparse code reveals the relationship between a data

sample and the dictionary atoms. We can preserve the relationship by fixing the coefficients when the dictionary atoms are updated, such that the approximation error is reduced. This can be contrasted with the dictionary update of K-SVD, where the dictionary atom and the corresponding coefficients are updated together using an SVD procedure. Given the sparse codes and the weighting matrix, the dictionary update step can be expressed as

$$\hat{\mathbf{\Psi}} = \underset{\mathbf{\Psi}}{\operatorname{argmin}} \sum_{i=1}^{T} \|\mathbf{y}_i - \mathbf{\Psi}\mathbf{z}_i\|_2, \text{ subject to } \|\boldsymbol{\psi}_k\|_2^2 = 1, \forall k = \{1, \ldots, K\}. \tag{9.11}$$

Here $\mathbf{z}_i = \mathbf{W}_i^{-1}\mathbf{x}_i$ denotes the reweighted coefficient vector obtained from the local sparse coding step. In order to update the k^{th} dictionary atom, we can simplify the objective function as

$$\begin{aligned} \left\| \mathbf{Y} - \sum_{j=1}^{K} \boldsymbol{\psi}_j \mathbf{z}_{j,r} \right\|_F^2 &= \left\| \mathbf{Y} - \left(\sum_{j \neq k} \boldsymbol{\psi}_j \mathbf{z}_{j,r} \right) - \boldsymbol{\psi}_k \mathbf{z}_{k,r} \right\|_F^2, \\ &= \|\mathbf{E}_k - \boldsymbol{\psi}_k \mathbf{z}_{k,r}\|_F^2. \end{aligned} \tag{9.12}$$

Here $\mathbf{z}_{k,r}$ denotes the row vector containing the set of coefficients of all training vectors corresponding to the dictionary atom $\boldsymbol{\psi}_k$. The objective can be rewritten as

$$\begin{aligned} \|\mathbf{E}_k - \boldsymbol{\psi}_k \mathbf{z}_{k,r}\|_F^2 &= tr[(\mathbf{E}_k - \boldsymbol{\psi}_k \mathbf{z}_{k,r})^T (\mathbf{E}_k - \boldsymbol{\psi}_k \mathbf{z}_{k,r})], \\ &= tr[\mathbf{E}_k^T \mathbf{E}_k - \mathbf{E}_k^T \boldsymbol{\psi}_k \mathbf{z}_{k,r} \\ &\quad -(\hat{\mathbf{z}}_{k,r})^T \boldsymbol{\psi}_k^T \mathbf{E}_k + (\mathbf{z}_{k,r})^T \boldsymbol{\psi}_k^T \boldsymbol{\psi}_k \mathbf{z}_{k,r}]. \end{aligned} \tag{9.13}$$

Since $\boldsymbol{\psi}_k^T \boldsymbol{\psi}_k = 1$, the optimization with respect to $\boldsymbol{\psi}_k$ will consider only the second and third terms. The dictionary atom $\boldsymbol{\psi}_k$ can therefore be updated as

$$\hat{\boldsymbol{\psi}}_k = \frac{\mathbf{E}_k (\mathbf{z}_{k,r})^T}{\|\mathbf{E}_k (\mathbf{z}_{k,r})^T\|_2}. \tag{9.14}$$

It can be observed that this dictionary update is equivalent to computing the weighted mean of the residual error vectors, where the weights are obtained using local sparse coding. Since the coefficients are fixed during the update, the dictionary atoms change slowly from the initial atoms over iterations.

9.3 Kernel Local Sparse Coding

It is well known that the kernel trick can be employed to capture the non-linear similarity of features. We propose to perform local sparse coding in a high dimensional feature space, in order to improve the discrimination power of the codes. Let us define a non-linear transformation to a feature space $\mathcal{F}$ as $\Phi : \mathbb{R}^M \mapsto \mathcal{F}$. We denote the set of training vectors $\mathbf{Y}$ in the feature space as $\Phi(\mathbf{Y})$. The kernel matrix $\mathbb{K} \in \mathbb{R}^{N \times N}$ is a Gram matrix of all the feature vectors, $\mathbb{K} = \Phi(\mathbf{Y})^T \Phi(\mathbf{Y})$. All computations in the feature space will be performed exclusively using kernel similarities. In this work, we consider the RBF kernel to perform local sparse coding. Given two data samples $\mathbf{y}_1$ and $\mathbf{y}_2$, the kernel similarity can be computed as

$$\mathbb{K}(\mathbf{y}_1, \mathbf{y}_2) = \exp(-\gamma(\|\mathbf{y}_1 - \mathbf{y}_2\|_2^2)). \tag{9.15}$$

The joint optimization of local sparse coding and dictionary learning can be performed in the feature space as

$$\left\{\hat{\mathbf{\Psi}}, \{\hat{\mathbf{x}}_i\}_{i=1}^T\right\} = \underset{\mathbf{\Psi}, \{\mathbf{x}_i\}_{i=1}^T}{\operatorname{argmin}} \sum_{i=1}^{T} \|\Phi(\mathbf{y}_i) - \Phi(\mathbf{\Psi})\mathbf{W}_i^{-1}\mathbf{x}_i\|_2^2 + \lambda\|\mathbf{x}_i\|_1, \tag{9.16}$$

with the constraint that $\|\mathbf{\Psi}_k\|_2 = 1, \forall k = \{1, \ldots, K\}$. Here,

$$\begin{aligned} \mathbf{W}_i(k,k) &= \|\Phi(\mathbf{y}_i) - \Phi(\boldsymbol{\psi}_{\boldsymbol{k}})\|_2^2 \\ &= \mathbb{K}(\mathbf{y}_i, \mathbf{y}_i) + \mathbb{K}(\boldsymbol{\psi}_{\boldsymbol{k}}, \boldsymbol{\psi}_{\boldsymbol{k}}) - 2\mathbb{K}(\mathbf{y}_i, \boldsymbol{\psi}_{\boldsymbol{k}}) \\ &= 2(1 - \mathbb{K}(\mathbf{y}_i, \boldsymbol{\psi}_{\boldsymbol{k}})). \end{aligned} \tag{9.17}$$

For each data sample $\mathbf{y}_i$, kernel local sparse coding can be performed by simplifying the objective function in (9.16) as

$$\hat{\mathbf{x}}_i = \underset{\mathbf{x}_i}{\operatorname{argmin}}\, \mathbf{x}_i^T \mathbf{W}_i^{-1} \mathbb{K}(\mathbf{\Psi}, \mathbf{\Psi}) \mathbf{W}_i^{-1} \mathbf{x}_i - 2\mathbb{K}(\mathbf{y}_i, \mathbf{\Psi}) \mathbf{W}_i^{-1} \mathbf{x}_i + \lambda\|\mathbf{x}_i\|_1 \tag{9.18}$$

This can be solved using the feature-sign search algorithm, with the additional complexity of evaluating the kernel matrices.

9.3.1 Dictionary Learning

In order to perform dictionary update using the kernel matrices, fixing the sparse codes and denoting $\mathbf{z}_i = \mathbf{W}_i^{-1}\mathbf{x}_i$, (9.16) can be rewritten as

$$\min_{\boldsymbol{\Psi}} \sum_{i=1}^{T} \|\Phi(\mathbf{y}_i) - \Phi(\boldsymbol{\Psi})\mathbf{z}_i\|_2^2 \text{ subject to } \|\boldsymbol{\psi}_k\| \leq 1, \forall k. \tag{9.19}$$

The objective in (9.19) can be equivalently expressed as

$$f = \sum_{i=1}^{T} \left[1 + \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \mathbf{z}_i(k_1)\mathbf{z}_i(k_2)\mathbb{K}(\boldsymbol{\psi}_{k_1}, \boldsymbol{\psi}_{k_2}) - 2\sum_{k_1=1}^{K} \mathbf{z}_i(k_1)\mathbb{K}(\boldsymbol{\psi}_{k_1}, \mathbf{y}_i) \right]. \tag{9.20}$$

Since the objective function to be minimized involves the kernel matrices, we can iteratively update one dictionary atom at a time using the procedure described in [167]. In order to update the k^{th} dictionary atom, we differentiate the objective in (9.20) as

$$\frac{\partial f}{\partial \boldsymbol{\psi}_k} = -4\gamma \sum_{i=1}^{T} \left[\sum_{k_2=1}^{K} \mathbf{z}_i(k)\mathbf{z}_i(k_2)\mathbb{K}(\boldsymbol{\psi}_k, \boldsymbol{\psi}_{k_2})(\boldsymbol{\psi}_k - \boldsymbol{\psi}_{k_2}) - \mathbf{z}_i(k)\mathbb{K}(\boldsymbol{\psi}_k, \mathbf{y}_i)(\boldsymbol{\psi}_k - \mathbf{y}_i) \right]. \tag{9.21}$$

By setting the derivative to zero, we can estimate the modified dictionary element. However, the kernel matrices $\mathbb{K}(\boldsymbol{\psi}_k, \boldsymbol{\Psi})$ and $\mathbb{K}(\boldsymbol{\psi}_k, \mathbf{Y})$ are unknown and hence we resort to using $\boldsymbol{\psi}_k^{(n-1)}$ from the earlier $(n-1)^{\text{th}}$ iteration of the algorithm to compute those kernel matrices at the n^{th} iteration. The dictionary atom update in the n^{th} iteration by solving

$$\sum_{i=1}^{T} \left[\sum_{k_2=1}^{K} \mathbf{z}_i(k)\mathbf{z}_i(k_2)\mathbb{K}(\boldsymbol{\psi}_k^{(n-1)}, \boldsymbol{\psi}_{k_2})(\boldsymbol{\psi}_k - \boldsymbol{\psi}_{k_2}) - \mathbf{z}_i(k)\mathbb{K}(\boldsymbol{\psi}_k^{(n-1)}, \mathbf{y}_i)(\boldsymbol{\psi}_k - \mathbf{y}_i) \right] = 0. \tag{9.22}$$

The solution is given as

$$\boldsymbol{\psi}_k^{(n)} = \frac{\boldsymbol{\Psi} diag[\mathbb{K}(\boldsymbol{\psi}_k(n-1), \boldsymbol{\Psi})]\mathbf{Z}\boldsymbol{z}_{k,r}^T - \mathbf{Y} diag[\mathbb{K}(\boldsymbol{\psi}_k^{(n-1)}, \mathbf{y})]\boldsymbol{z}_{k,r}^T}{\mathbb{K}(\boldsymbol{\psi}_k^{(n-1)}, \boldsymbol{\Psi})\mathbf{Z}\boldsymbol{z}_{k,r}^T - \mathbb{K}(\boldsymbol{\psi}_k^{(n-1)}, \mathbf{y})\boldsymbol{z}_{k,r}^T}. \tag{9.23}$$

Here $\boldsymbol{z}_{k,r}$ is a row vector containing the set of coefficients of all training vectors corresponding to the dictionary atom $\boldsymbol{\psi}_k$, and $diag[.]$ creates a diagonal matrix using the argument vector as its diagonal.

9.4 Object Recognition Experiments

In this section, we evaluate the performance of the proposed dictionaries in object recognition. Following the approach described in [100], we compute sparse codes for the local descriptors in an image and aggregate them at multiple spatial scales. In our simulations, all images are divided into patches of size 24×24 with a grid spacing of 8 and one SIFT descriptor is extracted per patch. The descriptors are coded using the following approaches: (a) regular sparse coding using K-means dictionary (SC) [100], (b) kernel sparse coding (KSC) [167], (c) Laplacian sparse coding (Laplacian SC) [160], (d) locality-constrained linear coding using K-means dictionary (LLC) [154], (e) local sparse coding using the proposed dictionary (LSC), and (f) kernel local sparse coding using the proposed dictionary (KLSC). We then employ spatial pyramid coding, where we generate region-specific codes at different spatial scales. Each image is processed at spatial scales 1, 2 and 4 respectively. In the first scale, where the full image is considered, the sparse codes are stacked into a matrix and max-pooling [154] is performed, where the row-wise maximum coefficient value is chosen, resulting in a vector. Note that we consider the absolute values for computing maximum at each index, but retain the sign. At scale 2, the image is split into 4 regions and 4 max pooled feature vectors are generated. Similarly, 16 feature vectors are obtained at scale 4. Finally, all the 21 feature vectors are stacked into a single column vector and used as the training feature to a linear SVM. We evaluated the object recognition performance using Caltech-256, UIUC sports, Corel-10 and Scene-15 datasets. We fixed the number of dictionary elements $K = 1024$ in all cases and randomly chose $75,000$ patches to generate the dictionary for each dataset. We set the parameter $\lambda = 0.3$ for the feature-sign search algorithm. For the parameter γ in the Gaussian kernel function, we selected the values $\frac{1}{64}, \frac{1}{128}, \frac{1}{128}, \frac{1}{256}$ on Scene-15, UIUC sports, Caltech-256 and Corel-10 respectively. It is evident from all the results presented in

Table 9.1: Comparison of the classification accuracies on the Caltech-256 dataset.

Method	**# Training Vectors**			
	15	**30**	**45**	**60**
SC [100]	27.73	34.02	37.46	40.14
KSC [167]	29.77	35.67	38.61	40.3
Laplacian SC [160]	30.01	35.74	38.54	40.43
LLC [154]	30.08	35.68	38.59	40.47
LSC (Proposed)	**30.19**	**35.94**	**38.72**	**40.66**
K-LSC (Proposed)	**30.28**	**36.05**	**38.84**	**40.82**

this section that the proposed algorithms achieve improved classification accuracies in comparison to the other approaches. All results presented were averaged over 10 iterations and the training/testing sets were randomly chosen in each iteration.

9.4.1 Caltech-256

The Caltech-256 dataset [179] contains $30,607$ images in 256 categories and its variability makes it extremely challenging in comparison to the Caltech-101 dataset. The intra-class variance is quite high and the objects of interest are not necessarily in the center of the images. We tested the performance of the different algorithms with 30, 45, and 60 training images per class respectively. Table 9.1 shows the classification accuracies obtained using the different algorithms. As it can be observed, kernel local sparse coding algorithm outperformed the other sparse coding based approaches in all cases.

9.4.2 Corel-10 Dataset

The Corel-10 dataset [222] contains 1000 images belonging to 10 categories: *skiing, beach, buildings, tigers, owls, elephants, flowers, horses, mountains* and *food*. Each class contains 100 images and we randomly select 50 images from each class for training. The results obtained are presented in Table 9.2.

Table 9.2: Comparison of the classification accuracies on the Corel-10, Scene-15, and UIUC sports datasets.

Method	**Accuracy (%)**		
	Corel-10	**Scene-15**	**UIUC Sports**
SC [100]	86.2	80.28	82.74
KSC [167]	89.43	83.08	84.92
Laplacian SC [160]	88.4	89.75	85.31
LLC [154]	88.41	89.78	85.34
LSC (Proposed)	**88.8**	**89.94**	**85.56**
K-LSC (Proposed)	**89.71**	**90.61**	**85.93**

9.4.3 Scene-15 Dataset

The Scene-15 dataset [99] contains 4485 images belonging to 15 different categories and is typically used for evaluating scene classification performance. The number of images per class vary between 200 and 400 images. The scene categories of this dataset include *suburb, coast, forest, highway, inside city, mountain, open country, street, tall building, office, bedroom, industrial, kitchen, living room* and *store*. We train the linear SVM by using features extracted from 100 images of each class use the rest for testing.

9.4.4 UIUC Sports Dataset

The UIUC sports dataset [223] contains 1792 images from 8 different classes: *badminton, bocce, croquet, polo, rock climbing, rowing, sailing* and *snow boarding*. The number of images in each class varies from 137 to 250. Following the standard evaluation procedure [167], we randomly select 70 images from each class for training and the rest for testing. The object recognition performance is reported in Table 9.2.

9.5 Image Retrieval

In this section, we present a supervised coding approach for performing image retrieval using sub-image heterogeneous features. In sparse coding based object recognition, we considered dense descriptors extracted from small patches in an image. However, prior knowledge of the labels/tags associated with the training images are not considered

when building the SPM feature. Incorporating supervised information into local sparse coding will result in highly discriminative features for retrieval. Hence, we propose to extract features from reasonably large regions in an image, since we can assign labels to those regions. Preliminary results for the proposed image retrieval framework were reported in [46].

9.5.1 Extracting Sub-image Features

In the proposed approach, we first divide an image into highly overlapping sub-images, that are much larger than image patches used for extracting local descriptors. For example, in an image of size 256×256, the sub-images can be of size 128×128. Since the sub-images are large, they correspond to significant portions of objects/scenes in the image. We extract multiple global/local features from each sub-image and build a heterogeneous feature by normalizing each feature to unit ℓ_2 norm, and stacking them together.

Designing a dictionary, for local sparse coding, using the heterogeneous features will enable us to identify key representative sub-image features found across all images. Such a dictionary captures the interactions between the objects and the background in the images and hence the resulting codes may retrieve more meaningful neighbors. This dictionary corresponds to a "bag of visual phrases" where each dictionary element corresponds to a visual phrase. This can be contrasted with the "bag of visual words", which is the dictionary that we would obtain if we had considered small image patches. Local sparse codes for the sub-image features can be obtained using the procedure in Section 9.2. The codes of all sub-images in an image are then aggregated in a spatial pyramid as described earlier, where the aggregation is performed using max-pooling. The resulting feature vector represents the importance of each visual phrase in the image. In Figure 9.1, we show one sample image each from three classes of the Microsoft Research Cambridge database [6]. The feature

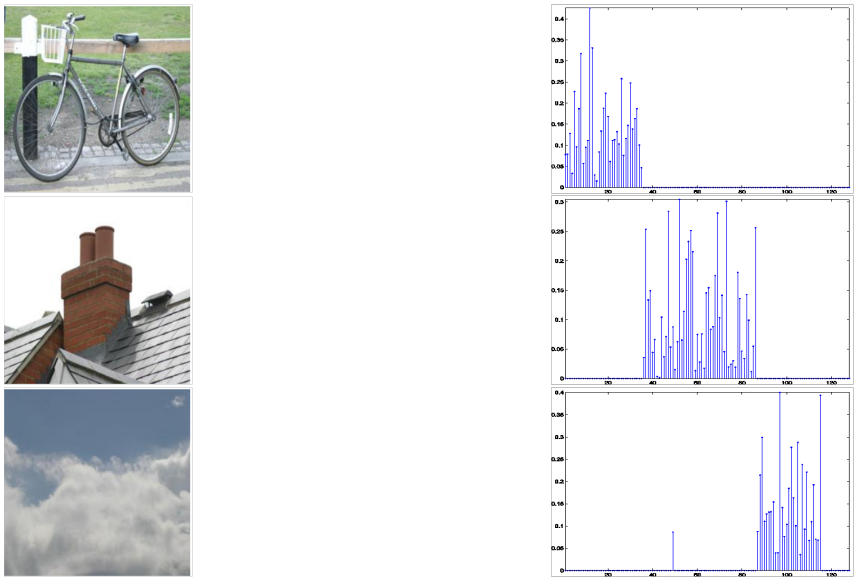


Figure 9.1: Sample images from the Microsoft Research Cambridge database [6] (left) and their aggregated sub-image features (right).

vectors obtained by aggregating the local sparse codes of sub-images at one spatial scale are shown on the right. In the first image, there is predominantly only one object present in the image (*bicycle*). The second image (*chimney*) contains geometric structures (walls, rooftops) in addition to the background (sky). In the third image, all sub-images are quite similar to cloud-like patterns. The feature vectors for the images illustrated have disjoint non-zero supports and this leads to high discrimination among classes.

9.5.2 Supervised Local Sparse Coding

As described earlier, we propose to employ a coding scheme that exploits the fact that once a dictionary atom has been chosen to represent a sub-image feature in a class, it may as well be used to represent other sub-image features of the same class. In addition to exploiting the class label information, this will ensure that aggregating the sparse codes from all sub-images will result in a sparse feature. In [159], the authors proposed a mixed-norm regularization method to perform group sparse coding of visual descriptors in an image. Furthermore, simultaneous sparse coding has been

successfully employed to obtain sparse codes for a set of data samples. In our setup, we assume that all sub-image features in a class can be modeled using the same neighborhood. In other words, we want to exploit the clusterability of the features within a class. For each class, the local neighborhood is identified by computing a representative feature for that class and then determining the weights corresponding to all dictionary atoms. If we employ the distance metric in (9.2), mean of all features is used as the representative. Whereas, the rank-1 SVD of the features is used as the representative when (9.3) is employed. Given the neighborhood, we perform supervised coding of the sub-image features in a class as

$$\hat{\mathbf{X}}^{(g)} = \underset{\mathbf{X}}{\operatorname{argmin}} \|\mathbf{Y}^{(g)} - \mathbf{\Psi}\mathbf{W}_{(g)}^{-1}\mathbf{X}\|_F^2 \text{ subject to } \|\mathbf{X}\|_{row-0} \leq L, \tag{9.24}$$

where $\mathbf{Y}^{(g)}$ is the set of training sub-image features in class g, $\mathbf{W}_{(g)}$ is the diagonal weight matrix corresponding to the estimated neighborhood for that class and $\|\mathbf{X}\|_{row-0}$ is the row-ℓ_0 pseudo norm of the coefficient matrix i.e., $\|\mathbf{X}\|_{row-0} = |\text{rowsupp}(\mathbf{X})|$. Here,

$$\text{rowsupp}(\mathbf{X}) = \{k \in \{1, \ldots, K\} : \mathbf{x}_{k,t} \neq 0 \text{ for some } t\}. \tag{9.25}$$

Several algorithms exist to solve this simultaneous sparse approximation problem [224] and any algorithm can be chosen depending on the accuracy and computational complexity requirements of the system.

The dictionary update step in Section 9.2.1 can be extended to the case of supervised local sparse coding straightforwardly. The objective function to be minimized when updating the dictionary atom $\boldsymbol{\psi}_k$ can be written as

$$\sum_{g=1}^{G} \left\| \mathbf{Y}^{(g)} - \mathbf{\Psi}\mathbf{W}_{(g)}^{-1}\mathbf{X}^{(g)} \right\|_F^2 \tag{9.26}$$

where G denotes the total number of classes. Substituting $\mathbf{Z}^{(g)} = \mathbf{W}_{(g)}^{-1}\mathbf{X}^{(g)}$, the

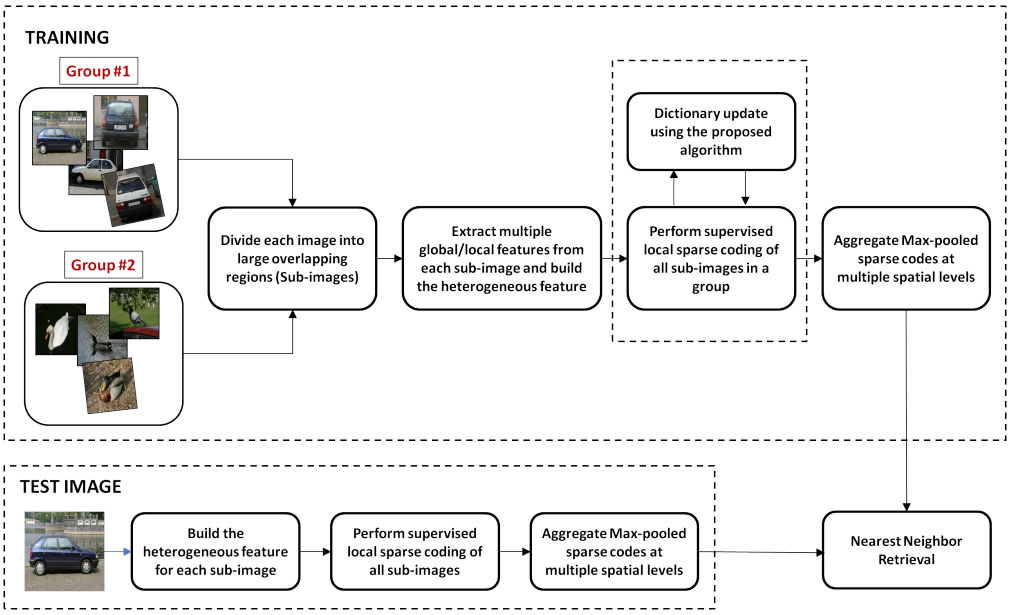


Figure 9.2: Proposed supervised local sparse coding algorithm for image retrieval using sub-image heterogeneous features.

objective can be simplified as

$$\sum_{g=1}^{G} \left\| \mathbf{Y}^{(g)} - \boldsymbol{\Psi}\mathbf{Z}^{(g)} \right\|_F^2 = \sum_{g=1}^{G} \left\| \mathbf{E}_k^{(g)} - \boldsymbol{\psi}_k \mathbf{z}_{k,r}^{(g)} \right\|_F^2 \tag{9.27}$$

$$= \left\| \left[\mathbf{E}_k^{(1)} \dots \mathbf{E}_k^{(G)} \right] - \boldsymbol{\psi}_k \left[\mathbf{z}_{k,r}^{(1)} \dots \mathbf{z}_{k,r}^{(G)} \right] \right\|_F^2$$

$$= \left\| \tilde{\mathbf{E}}_k - \boldsymbol{\psi}_k \tilde{\mathbf{z}}_k \right\|_F^2, \tag{9.28}$$

where $\mathbf{E}_k^{(g)} = \mathbf{Y}^{(g)} - \sum_{j \neq k} \boldsymbol{\psi}_j \mathbf{z}_{j,r}^{(g)}$ and $\boldsymbol{\psi}_k$ is constrained to be unit norm. In effect, the updated dictionary atom can be obtained using the procedure in (9.14) with the concatenated error matrix $\tilde{\mathbf{E}}_k$ and the coefficient vector $\tilde{\mathbf{z}}_k$ (corresponding to $\boldsymbol{\psi}_k$).

9.5.3 Simulations

Figure 9.2 illustrates the proposed algorithm for performing image retrieval using sub-image features. In our simulations, we obtained images from the Microsoft Research Cambridge image database [6] and resized them to 256×256. The dataset consists of $4,323$ images belonging to 19 different classes. We used 75% of images randomly chosen from each class for training and the rest for testing. For each image, we extracted

the following features: *Histogram of Oriented Gradients* (HoG), *GIST* and *Local Binary Pattern* (LBP). We obtained the heterogeneous base feature for each image by ℓ_2 normalizing and stacking the three features. To generate the sub-image features, we divided each image into overlapping patches of size 128×128 (grid spacing: 16) and obtained the base feature for each patch. We employed the algorithm in Section 9.5.2 to learn a dictionary containing 512 atoms using sub-image features from the training set. The supervised local sparse codes from each image were aggregated in 2 spatial scales. In the first spatial level, all sub-image features were pooled together to form a 512×1 vector. In spatial level 2, each image was divided into 4 regions and sparse codes of only sub-images from that region were aggregated. Aggregated codes from both levels were stacked to form a 2560×1 vector and ℓ_2 normalized. For each test image, all sub-image features were simultaneously coded using (9.24) and spatial pyramid features were constructed.

For comparison we generated spatial pyramid features by aggregating local sparse codes obtained by not incorporating the class label information. We employed a simple Euclidean distance based nearest neighbor search on the features to identify visually similar images. Figure 9.3 demonstrates the Precision vs Recall curves, obtained using the original heterogeneous features, proposed local sparse coded sub-image features with/without label information for 5 different classes from the dataset. Note that the curves are obtained by averaging the retrieval results of all test images in every class. As it can be observed, in all cases both local sparse coding based features performed significantly better than the heterogeneous features. Furthermore, the performance improvement by employing supervised coding is also evident from the results.

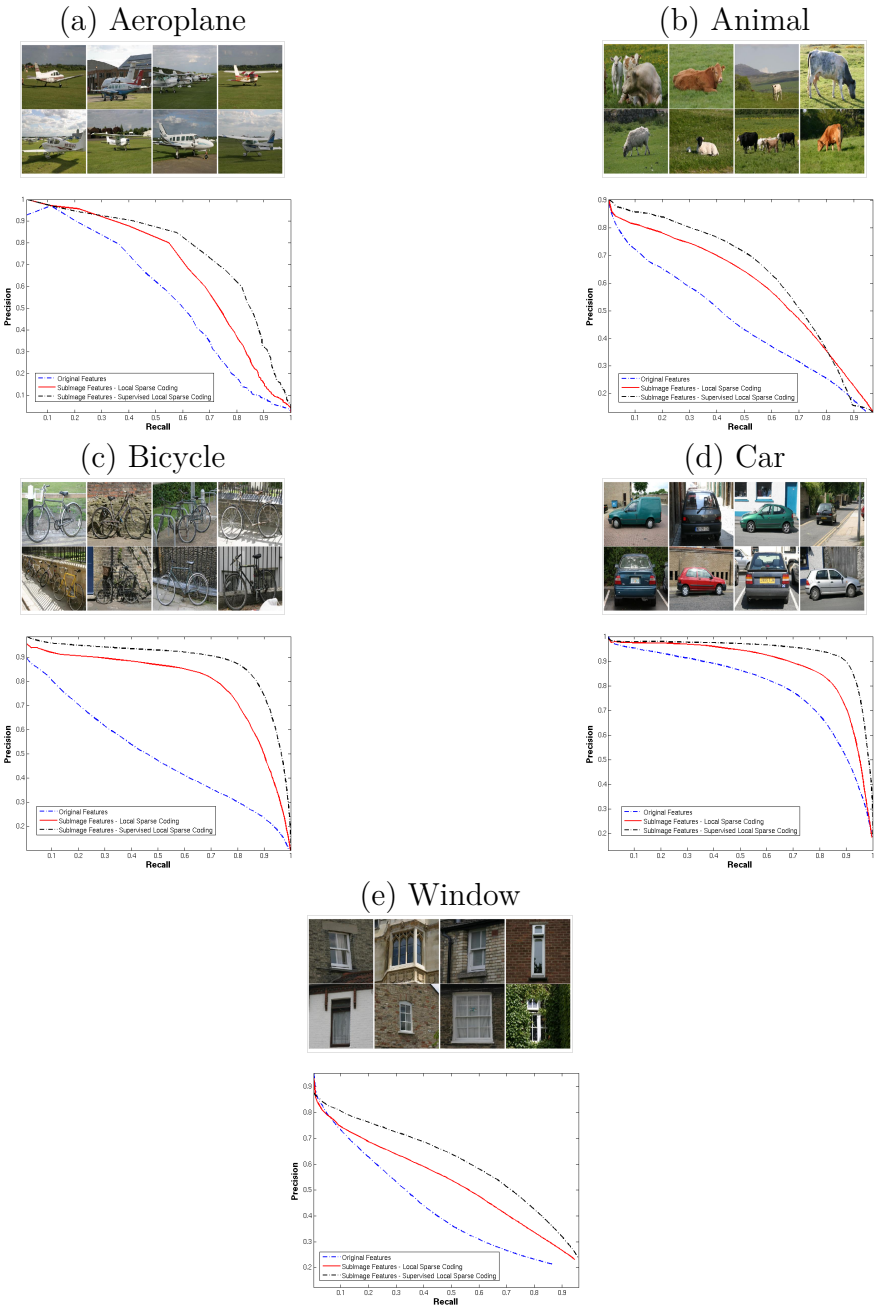


Figure 9.3: Precision vs Recall curves for different classes from the Microsoft Research Cambridge image database [6]. In each case, a sample set of test images are shown. The images are best viewed in color and 300% zoom-in.

Chapter 10

SUMMARY AND FUTURE WORK

10.1 Summary

In this dissertation, different aspects of sparse learning were considered, and a range of algorithms was presented to use sparse models in image understanding and computer vision applications. Though sparse methods have been used in large scale learning, because of their ability to provide parsimonious models, they can be made more effective by incorporating ideas from statistical learning theory and machine learning. Inferring dictionaries for sparse representations using training examples is a well-studied problem. These methods have been adapted to several supervised and unsupervised learning problems by incorporating a variety of discriminatory constraints, and learning strategies. However, by considering the characteristics of dictionary learning algorithms, from the perspective of statistical learning, more effective image models can be derived. Furthermore, modern tools from machine learning such as kernel methods, ensemble methods, graph-embedding approaches, locally-linear methods can enrich the use of sparse modeling paradigm in computer vision applications. In the rest of this section, a detailed summary of the findings in this dissertation is provided.

To facilitate the understanding of dictionary learning algorithms, the problem of 1-D subspace clustering was considered, and its theoretical and algorithmic characteristics were analyzed. Since dictionary learning is a generalization of 1-D subspace clustering (K-lines), the findings from this analysis can be highly beneficial in improving the behavior of sparse learning algorithms. The convergence and local optimality of this clustering procedure was proved, and an Expectation-Maximization formulation was developed to show that dictionary learning is indeed a generalization of 1-D subspace clustering. Though performance of a clustering algorithm is often determined by a suitable initialization, its stability characteristics can be used to evaluate

the clusters identified by the algorithm. It was proved that the K-lines clustering algorithm is stable under certain conditions. In spite of the markedly different geometries of the K-lines and the K-means clustering algorithms, they both are locally optimal and their distortion functions belong to the uniform Donsker class which leads to similar stability properties.The overhead of computing SVD for high dimensional data in K-lines clustering was addressed. Different approaches for dimensionality reduction based on random projections were presented and applied to cluster video data. Performing clustering in a feature space, obtained using non-linear transformations, can improve the clustering performance because of the improved separability between the classes. Hence, the kernel K-lines clustering algorithm, that uses a simple iterative procedure for identifying 1-D subspaces, was developed.

Though clustering algorithms are unsupervised, and hence their performance is dependent on the class separability, performing discriminative clustering can improve their performance. To incorporate discriminatory information, approaches such as linear discriminant analysis (LDA) are used to identify a low-dimensional subspace, where clustering is performed. The label information for LDA can be obtained from the previous iteration of clustering. Since 1-D subspace clustering can be very useful for estimating the unknown mixing process in blind source separation, an algorithm to perform discriminative K-lines clustering was developed. The method works by iteratively identifying projections to discriminate the observations from different sources, and computing the mixing parameters using 1-D subspace clustering. The proposed algorithm can work in all configurations of mixing, relaxes the dependency on source sparsity and disjointedness, and can refine the rough estimate of the number of sources provided. For overdetermined and fully determined mixing conditions, discrimination is improved using appropriate dimensionality reduction. Since it is not possible to reduce dimensionality in underdetermined mixing, the algorithm uses a novel feature space method to overcome this problem. Simulation results with

speech mixtures indicate that the proposed technique achieves improved estimation when compared to state-of-the-art approaches, and enables a robust estimation of the number of sources. The proposed method can have a significant impact in the separation of speech and acoustic mixtures in hearing aid design, and extraction of artifacts in biomedical signals.

Sparse learning amounts to building a suitable dictionary that can efficiently represent the training data samples. The performance of such a dictionary is often measured by its ability to represent novel test samples, not present in the training set. By inferring dictionaries that can model the entire data space, and not just the samples considered, *global* dictionaries can be designed for large scale problems. A multilevel learning algorithm to design global dictionaries that exploit the redundancy and energy hierarchy found in natural image patches. The proposed algorithm employs K-lines clustering to learn atoms for each level of the dictionary. It was showed that the algorithm converges for a sufficient number of levels and that energy hierarchy is exhibited for a sufficient number of atoms per level. In addition, theoretical aspects of learning such as stability and generalization were considered. It was proved that the dictionaries learned using different sets of training data, from the same probability space, are arbitrarily close to each other, for a sufficiently large number of data samples. Furthermore, the asymptotic generalization characteristics was analyzed, and the stability and generalization behavior were demonstrated using simulations. Finally, to improve the generalization capability of multilevel dictionaries, it was proposed to perform ensemble learning in each level of multilevel dictionary design. Simulation results for compressed recovery, image denoising, and image compression clearly demonstrated that the proposed multilevel dictionaries provide superior performance when compared to global K-SVD dictionaries.

Inferring sparse models in a feature space provides a flexible way to model nonlinear relationship between the training examples, which is crucial for discrimination

tasks such as object recognition. Using kernel methods to address this problem will allow the use of multiple features to identify the similarity between two images. The idea of multiple kernel sparse models was developed to enable the design of sparse models in a unified space obtained using multiple kernels. The proposed approaches are of importance in complex vision problems, wherein they effectively characterize the image data, by fusing the information obtained from a large set of visual features. By learning sparse models in the multiple kernel domain, compact representations were obtained for the images and discrimination was achieved by exploiting the interactions between the multiple descriptors. Two different approaches for obtaining MKSR and optimizing the dictionaries for the feature space were developed. The proposed algorithms were comprehensively evaluated in supervised object recognition and unsupervised clustering tasks and the simulation results demonstrate the effectiveness of the proposed MKSR algorithms.

Since kernel methods allow the computation of sparse codes for diverse features, multiple kernel sparse representations can be effective in image segmentation. A novel, automated segmentation technique for detecting brain tumors was proposed in this dissertation. In the new approach, ensemble kernel matrices were constructed using using the pixel intensities and their spatial locations, and kernel dictionaries were designed for sparse coding pixels in a non-linear feature space. The resulting sparse codes were used to train a linear SVM classifier that determines if a pixel in the image belongs to an active tumor region. Furthermore, a semi-automated segmentation approach was proposed that uses two kernel dictionaries to model the tumor and non-tumor pixels respectively, and employs a simple error-based classifier. Us- ing simulations on a real dataset obtained for 9 different patients, it was demonstrated that both of the proposed approaches resulted in accurate tumor identifications in comparison to the widely used tumor segmentation approaches.

Local sparse modeling can be effective in modeling samples lying close to a manifold, which is true in several classification problems. In this dissertation, dictionary learning algorithms were proposed to perform local sparse coding of image features for object recognition and image retrieval. Since the kernel trick can be used to exploit the non-linear similarity of the features, local sparse coding in a high-dimensional feature space obtained by an implicit mapping function. When applied for object recognition, the proposed algorithms achieved im460 proved recognition performance in comparison to other sparse coding based classification approaches. Incorporating supervised (label/tag) information when coding sub-image heterogeneous features for image retrieval, resulted in highly discriminative local sparse codes. Using simulations, improvements in image retrieval performance by using supervised local sparse coding were demonstrated.

Measuring the number of glomeruli in the entire, intact kidney using non-destructive techniques is of immense importance in studying several renal and systemic diseases. Commonly used techniques either require destruction of the entire kidney or perform extrapolation from measurements obtained from a few isolated sections. A recent magnetic resonance imaging (MRI) method, based on the injection of a contrast agent (cationic ferritin), can be used to effectively identify glomerular regions in the kidney. In this paper, we propose a robust, accurate, and low-complexity method for estimating the number of glomeruli from such kidney MRI images. The proposed technique uses a few expert-marked training images to obtain discriminative graphs that can distinguish the glomerular regions from the rest. These graphs are obtained using non-local sparse coding-based approaches and discriminative projection directions are obtained using graph-embedding techniques. For novel test images, patches are extracted and embedded in a low-dimensional space using the discriminative projection directions. Glomerular regions are identified and counted by clustering the embedded test data. Results with experimental data show that

the proposed algorithm performs accurate glomerular counting, when compared to existing approaches, at a near real-time speed.

10.2 Future Work

The paradigm of sparse representations has enabled the design of mathematical models that can describe the lower level processing in the primary visual cortex of the human brain. Furthermore, performing suitable non-linear aggregation of these low-level features has resulted in efficient recognition frameworks. Though scientists are far from providing a comprehensive description of the human visual system, incorporating several learning aspects from human vision allows us to build biologically possible models for image understanding and computer vision. In addition, addressing the challenges in translating these ideas to prototype engineering systems is of immense importance. Building scalable models for large scale is a more contemporary challenge that adds a novel dimension to sparse learning. These requirements present huge potential in extending the research presented in this dissertation.

Learning sparse models in a large scale setting calls for the design of several theoretical and algorithmic tools. Incorporating strategies from ensemble learning improved the performance of global multilevel dictionaries in image restoration problems. Understanding the impact of these ensemble methods on the generalization characteristics of learned dictionaries will enable to build effective application-specific dictionaries. In particular, deriving generalization bounds for the finite sample case will be valuable for implementing sparse methods in real-world systems. The other important aspect that needs to be considered is the learning efficiency of these models. Though the dictionary training is performed online, it is often infeasible to learn models using millions of training samples. Hence, algorithms that can perform online learning in order to optimize the sparsifying dictionaries can be designed. From a statistical viewpoint, we need to compute an initial empirical estimate of the under-

lying parameters, and improve the estimates with adding more training examples in batches such that the empirical estimates approach their expected values. By understanding online learning methods from the viewpoint of algorithmic stability can also lead to improved global models. Furthermore, realizing efficient implementations using the computational resources of the CPU and the GPU (Graphical Processing Unit) will be an important step towards large scale learning. Several optimization strategies from supervised and unsupervised learning can be adopted to these global dictionaries.

Approaches for building structured dictionaries in order to achieve improved regularization can also be considered. Instead of learning a single dictionary for representing all image patches, building multiple dictionaries of varying complexity can be beneficial in a variety of ill-conditioned image recovery problems. Though quantification of complexity can be carried out based on several information theoretic principles, sparsity is a natural choice. Hence, image reconstruction amounts to first performing model selection from the set of models of varying complexity, such that it will result in the sparsest solution. In particular a hierarchical learning framework can be designed where in addition to imposing sparsity on the image representation, we constrain the dictionary atoms to be sparse in the dictionaries lower in the hierarchy. This is referred to as *Double Sparsity*. Such an algorithm will ensure that any dictionary atom can be sparsely decomposed into a set of other dictionaries. Following this, we need to learn statistical models that computes the likelihood of generating an image patch, by sparse coding, using each of the dictionaries. This will provide a principled way of identifying the complexity of the patch and choosing the suitable model for efficiently representing it. This approach can be very useful in recovery problems such as image denoising, image super-resolution, and image inpainting.

The analysis of K-lines clustering revealed that the distortion function, though does not belong to the class of Bregman divergences, can lead to stable clusterings.

However, several forms of distortion (or) loss functions used in machine learning are often based on a Bregman divergence. Hence, building a set of analysis tools based on statistical learning theory to understand the stability characteristics of such distortion functions will be very valuable in several supervised and unsupervised learning methods.

The use of sparse models in recognition applications can be made more effective by incorporating principles from human vision, and modern machine learning. Kernel methods were found to be particularly useful in exploiting the non-linear relation between the features, and fusing multiple descriptors to describe the image data. Performing theoretical analysis of kernel sparse learning in comparison to kernel support vector machines can provide new insights for improving the discrimination power of sparse methods. In recent years, there have been significant fundamental advances in understanding differential geometric properties of several features and models with wide applications such as shape analysis, face recognition, activity recognition, visual detection and tracking. Some of the challenges in performing machine learning tasks over such non-Euclidean spaces is to effectively exploit the intrinsic geometric properties of the manifold of interest. In recent years, the knowledge of geometric properties such as tangent-spaces, exponential maps, inverse exponential maps etc. have been utilized to devise sophisticated algorithms for tasks such as classification, clustering, and retrieval. By designing dictionary learning algorithms for these non-Euclidean manifolds, sparse models can be made very effective in high-dimensional data visualization.

REFERENCES

[1] "Berkeley segmentation dataset," Available at http://www.eecs.berkeley.edu/ Research/Projects/CS/vision/grouping/segbench/.

[2] B. Olshausen, "Sparsenet matlab toolbox," Available at http://redwood.berkeley.edu/bruno/sparsenet/.

[3] V. G. Reju, S. Koh, and Y. Soon, "An algorithm for mixing matrix estimation in instantaneous blind source separation," *Signal Processing*, vol. 89, pp. 1762–1773, 2009.

[4] "K-SVD matlab toolbox," Available at http://www.cs.technion.ac.il/ elad/software/.

[5] "USPS dataset," Available at ftp://ftp.kyb.tuebingen.mpg.de/pub/bs/data/.

[6] I. Ulusoy and C. M. Bishop, "Generative versus discriminative methods for object recognition," in *IEEE CVPR*, 2005.

[7] D. Field, "Relations between the statistics of natural images and the response properties of cortical cells," *Journal of the Optical Society of America*, vol. 4, pp. 2379–2394, 1987.

[8] E. Simoncelli, "Modeling the joint statistics of images in the wavelet domain," *Proceedings of the SPIE 44th Annual Meeting*, July 1999.

[9] A. B. Lee, K. S. Pedersen, and D. Mumford, "The nonlinear statistics of high-contrast patches in natural images," *International Journal of Computer Vision*, vol. 54, no. 1, pp. 83–103, 2003.

[10] G. C. et. al., "On the local behavior of spaces of natural images," *International Journal of Computer Vision*, vol. 2007, 2006.

[11] D. Graham and D. Field, "Natural images: Coding efficiency," *Encyclopedia of Neuroscience*, vol. 6, pp. 19–27, 2008.

[12] S. Marčelja, "Mathematical description of the responses of simple cortical cells*," *JOSA*, vol. 70, no. 11, pp. 1297–1300, 1980.

[13] J. Jones and L. Palmer, "An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex," *Journal of Neurophysiology*, vol. 58, no. 6, pp. 1233–1258, 1987.

[14] D. Field and D. Tolhurst, "The structure and symmetry of simple-cell receptive-field profiles in the cat's visual cortex," *Proceedings of the Royal society of London. Series B. Biological sciences*, vol. 228, no. 1253, pp. 379–400, 1986.

[15] D. Chandler and D. Field, "Estimates of the information content and dimensionality of natural scenes from proximity distributions," *JOSA A*, vol. 24, no. 4, pp. 922–941, 2007.

[16] M. Riesenhuber and T. Poggio, "Neural mechanisms of object recognition," *Current opinion in neurobiology*, vol. 12, no. 2, pp. 162–168, 2002.

[17] D. J. Field, "What is the goal of sensory coding?" *Neural Computation*, vol. 6, pp. 559–601, 1994.

[18] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision Research*, vol. 37, no. 23, pp. 3311–3325, December 1997.

[19] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek, "An overview of jpeg-2000 (2000)," in *Proceedings of IEEE Data Compression Conference*, Snowbird, Utah, 2000, pp. 523–541.

[20] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing.* Springer, 2010.

[21] Y. C. Eldar, P. Kuppinger, and H. Bolcskei, "Block-sparse signals: Uncertainty relations and efficient recovery," *IEEE Transactions on Signal Processing*, vol. 58, no. 6, pp. 3042–3054, 2010.

[22] J. Huang, T. Zhang, and D. Metaxas, "Learning with structured sparsity," *Proceedings of the 26th Annual International Conference on Machine Learning ICML 09*, pp. 1–8, 2009.

[23] Z. He *et.al.*, "K-hyperline clustering learning for sparse component analysis," *Signal Processing*, vol. 89, pp. 1011–1022, 2009.

[24] S. Ben-David, U. von Luxburg, and D. Pál, "A sober look at clustering stability," *Proceedings of the Conference on Computational Learning Theory*, pp. 5–19, 2006.

[25] S. Ben-David, D. Pál, and H. U. Simon, "Stability of K-means clustering," *Learning Theory*, vol. 4539, pp. 20–34, 2007.

[26] A. Rakhlin and A. Caponnetto, "Stability of K-means clustering," in *Advances in Neural Information Processing Systems*, B. Schölkopf, J. Platt, , and T. Hoffman, Eds., vol. 19. Cambridge, MA: MIT Press, 2007.

[27] G. Hinton, "Learning to represent visual input," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 365, no. 1537, pp. 177–184, 2010.

[28] H. Lee, C. Ekanadham, and A. Ng, "Sparse deep belief net model for visual area v2," *Advances in neural information processing systems*, vol. 20, pp. 873–880, 2008.

[29] C. MarcâĂŹAurelio Ranzato, S. Chopra, and Y. LeCun, "Efficient learning of sparse representations with an energy-based model," *Advances in neural information processing systems*, vol. 19, pp. 1137–1144, 2006.

[30] R. Rigamonti, M. Brown, and V. Lepetit, "Are sparse representations really relevant for image classification?" in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 1545–1552.

[31] M. Brown, G. Hua, and S. Winder, "Discriminative learning of local image descriptors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 1, pp. 43–57, 2011.

[32] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, "Robust object recognition with cortex-like mechanisms," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 3, pp. 411–426, 2007.

[33] M. Girolami, "Mercer kernel-based clustering in feature space," *IEEE Transactions on Neural Networks*, vol. 13, pp. 780–784, 2002.

[34] B. C. B. Cheng, J. Y. J. Yang, S. Y. S. Yan, Y. F. Y. Fu, and T. S. Huang, "Learning with ℓ_1-graph for image analysis," *IEEE Transactions on Image Processing*, vol. 19, no. 4, pp. 858–866, 2010.

[35] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Optimality and stability of the K-hyperline clustering algorithm," *Pattern Recognition Letters*, vol. 32, no. 9, pp. 1299–1304, 2011.

[36] J. J. Thiagarajan, K. N. Ramamurthy and A. Spanias, "Dimensionality reduction for distance based video clustering," in *Artificial Intelligence Applications and Innovations*, ser. IFIP Advances in Information and Communication Technology, vol. 339, 2010, pp. 270–277.

[37] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Mixing matrix estimation using discriminative clustering for blind source separation," *Digital Signal Processing*, 2012.

[38] J. J. Thiagarajan, K. N. Ramamurthy, A. Spanias, and P. Nasiopoulos, "Learning multilevel dictionaries for compressed sensing using discriminative clustering," in *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2012 Eighth International Conference on*. IEEE, 2012, pp. 494–497.

[39] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Multilevel dictionary learning for sparse representation of images," in *Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop (DSP/SPE), 2011 IEEE*. IEEE, 2011, pp. 271–276.

[40] ——, "Learning stable multilevel dictionaries using K-hyperline clustering," *IEEE Transactions on Neural Networks and Learning Systems*, 2013. [Online]. Available: http://arxiv.org/pdf/1303.0448v1.pdf

[41] ——, "Multiple kernel sparse representations for object recognition and unsupervised clustering," *IEEE Transactions on Image Processing*, 2013. [Online]. Available: http://arxiv.org/pdf/1303.0582v1.pdf

[42] J. Thiagarajan, D. Rajan, K. Ramamurthy, D. Frakes, and A. Spanias, "Automated tumor segmentation using kernel sparse representations," *2012 IEEE 12th International Conference on Bioinformatics and Bioengineering (BIBE)*, pp. 401–406, 2012.

[43] J. J. Thiagarajan, K. N. Ramamurthy, D. Rajan, A. Spanias, D. Frakes, and A. Puri, "Kernel sparse models for automated tumor segmentation," *International Journal on Artificial Intelligence Tools (Submitted)*, 2013.

[44] J. J. Thiagarajan, K. N. Ramamurthy, D. Frakes, and A. Spanias, "An algorithm to measure glomerular number in kidney MRI images," *IEEE Transactions on Biomedical Engineering (Submitted)*, 2013.

[45] J. J. Thiagarajan and A. Spanias, "Learning dictionaries for local sparse coding in image classification," in *Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference on.* IEEE, 2011, pp. 2014–2018.

[46] J. J. Thiagarajan, K. N. Ramamurthy, P. Sattigeri, and A. Spanias, "Supervised local sparse coding of sub-image features for image retrieval," in *IEEE ICIP*, 2012.

[47] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Local sparse coding for image classification and retrieval," *Eurasip Journal on Image and Video Processing (Submitted)*, 2013.

[48] R. Bracewell, *The Fourier Transform and Its Applications (3rd edition).* McGraw-Hill Science Engineering, 1999.

[49] I. Daubechies, *Ten Lectures on Wavelets.* SIAM, 1992.

[50] E. J. Candès and D. L. Donoho, "New tight frames of curvelets and optimal representations of objects with C^2 singularities," Department of Statistics, Stanford University, USA, Tech. Rep., 2002.

[51] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via l^1 minimization," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, no. 5, pp. 2197–2202, March 2003.

[52] A. Spanias, "A hybrid transform method for speech analysis and synthesis," *Signal Processing*, vol. 24, pp. 217–229, August 1991.

[53] J. A. Tropp, "Topics in sparse approximation," Ph.D. dissertation, University of Texas, Austin, 2004.

[54] B. D. Rao and Y. Bresler, "Signal processing with sparseness constraints," in *Proceedings of the 1998 IEEE International Conference on Acoustics,Speech and Signal Processing*, Seattle, 1998.

[55] P. Frossard and P. Vandergheynst, "Redundant representations in image processing," in *Proceedings of the 2003 IEEE International Conference on Image Processing*, Barcelona, Spain, 2003.

[56] R. R. Coifman and D. L. Donoho, "Translational invariant de-noising," Wavelets and Statistics, Lecture Notes in Statistics, Tech. Rep., 1995.

[57] D. Donoho, "Neighborly polytopes and sparse solution of underdetermined linear equations," Stanford University, Tech. Rep., 2005.

[58] J. Tropp, "On the conditioning of random subdictionaries," *Applied and Computational Harmonic Analysis*, vol. 25, no. 1, pp. 1–24, 2008.

[59] G. Davis, S. Mallat, and M. Avellaneda, "Greedy adaptive approximation," *Journal of Constructive Approximation*, vol. 13, pp. 57–98, 1997.

[60] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[61] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, October 2004.

[62] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Review*, vol. 43, no. 1, pp. 129–159, 2001.

[63] I. F. Gorodnitsky and B. D. Rao, "Sparse signal reconstruction from limited data using FOCUSS: A re-weighted norm minimization algorithm," *IEEE Transactions on Signal Processing*, vol. 45, no. 3, pp. 600–616, March 1997.

[64] H. Lee, A. Battle, R. Raina, and A. Ng, "Efficient sparse coding algorithms," *Advances in neural information processing systems*, vol. 19, p. 801, 2007.

[65] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[66] M. Elad, "Why simple shrinkage is still relevant for redundant representations?" *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5559 –5569, December 2006.

[67] M. Elad, B. Matalon, J. Shtok, and M. Zibulevsky, "A wide-angle view at iterated shrinkage algorithms," in *SPIE (Wavelet XII) 2007*, 2007.

[68] J. A. Tropp and A. Gilbert, "Signal recovery from partial information via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, 2005.

[69] J. Fuchs, "Recovery of exact sparse representations in the presence of bounded noise," *Information Theory, IEEE Transactions on*, vol. 51, no. 10, pp. 3601–3608, 2005.

[70] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[71] L. K. Jones, "On a conjecture of huber concerning the convergence of projection pursuit regression," *Annals of Statistics*, vol. 15, no. 2, pp. 880–882, 1987.

[72] S. Chen, S. A. Billings, , and W. Luo, "Orthogonal least squares methods and their application to nonlinear system identification," *International Journal of Control*, vol. 50, no. 5, pp. 1873–1896, 1989.

[73] G. Davis, S. Mallat, and Z. Zhang, "Adaptive time-frequency decompositions," *Optical Engineering*, vol. 33, no. 7, p. 2183ÂŰ2191, July 1994.

[74] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proceedings of 27th Annual Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, California, November 1993.

[75] S. Weisberg, *Applied linear regression*. Wiley, 2005, vol. 528.

[76] A. Bruckstein, M. Elad, and M. Zibulevsky, "On the uniqueness of nonnegative sparse solutions to underdetermined systems of equations," *IEEE Transactions on Information Theory*, vol. 54, no. 11, pp. 4813–4820, 2008.

[77] D. Needell and J. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 301–321, 2009.

[78] J. Starck, F. Murtagh, and J. Fadili, *Sparse image and signal processing: wavelets, curvelets, morphological diversity*. Cambridge Univ Pr, 2010.

[79] J. Starck, D. Donoho, and E. Candes, "Very high quality image restoration by combining wavelets and curvelets," in *Proc. SPIE*, vol. 4478, 2001, pp. 9–19.

[80] R. Gribonval and M. Nielsen, "Sparse representations in unions of bases," *IEEE Transactions on Information Theory*, vol. 49, no. 12, pp. 3320–3325, 2003.

[81] S. Mallat, *A wavelet tour of signal processing*. Academic press, 1999.

[82] K. N. Ramamurthy, J. J. Thiagarajan, and A. Spanias, "Improved sparse coding using manifold projections," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*. IEEE, 2011, pp. 1237–1240.

[83] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[84] K. Engan, S. O. Aase, and J. H. Husoy, "Method of optimal directions for frame design," in *Proceedings of IEEE ICASSP*, 1999.

[85] K. Engan, B. Rao, and K. Kreutz-Delgado, "Frame design using FOCUSS with method of optimal directions (MoD)," in *Norwegian Signal Processing Symposium*, 1999.

[86] I. F. Gorodnitsky and B. D. Rao, "Sparse signal reconstruction from limited data using focuss: A re-weighted norm minimization algorithm," *IEEE Transactions on Signal Processing*, vol. 45, no. 3, pp. 600–616, March 1997.

[87] S. F. Cotter, M. N. Murthi, and B. D. Rao, "Fast basis selection methods," in *Proceedings of 31st Annual Asilomar Conference on Signals, Systems and Computers*, vol. 2, Pacific Grove, California, November 1997.

[88] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 689–696.

[89] L. Bottou and O. Bousquet, "13 the tradeoffs of large-scale learning," *Optimization for Machine Learning*, p. 351, 2011.

[90] G. Yu, G. Sapiro, and S. Mallat, "Image modeling and enhancement via structured sparse model selection," in *Proc. of IEEE ICIP*, Sep. 2010, pp. 1641 –1644.

[91] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 2272–2279.

[92] K. Fukunaga, *Introduction to statistical pattern recognition (2nd ed.)*. Academic Press, 1990.

[93] K. Huang and S. Aviyente, "Sparse representation for signal classification," *East*, vol. 19, pp. 609–616, 2007.

[94] F. Rodriguez and G. Sapiro, "Sparse representations for image classification: Learning discriminative and reconstructive non-parametric dictionaries," *Security*, 2008. [Online]. Available: http://www.ima.umn.edu/preprints/jun2008/2213.pdf

[95] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Supervised dictionary learning," in *Proceedings of NIPS*, 2009.

[96] J. Wright *et.al.*, "Robust face recognition via sparse representation," *IEEE Trans. on PAMI*, vol. 31, no. 2, pp. 210–227, 2001.

[97] J. J. Thiagarajan, K. N. Ramamurthy, P. Knee, A. Spanias, and V. Berisha, "Sparse representations for automatic target classification in sar images," in

Communications, Control and Signal Processing (ISCCSP), 2010 4th International Symposium on. IEEE, 2010, pp. 1–4.

[98] R. Raina *et.al.*, "Self-taught learning: Transfer learning from unlabeled data," in *Proceedings of ICML*, 2007.

[99] S. Lazebnik *et.al.*, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," *Proceedings of CVPR*, 2006.

[100] J. Yang *et.al.*, "Linear spatial pyramid matching using sparse coding for image classification," in *Proceedings of IEEE CVPR*, 2009.

[101] P. Knee, J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Sar target classification using sparse representations and spatial pyramids," in *Radar Conference (RADAR), 2011 IEEE*. IEEE, 2011, pp. 294–298.

[102] P. Sattigeri, J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Implementation of a fast image coding and retrieval system using a gpu," in *Emerging Signal Processing Applications (ESPA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 5–8.

[103] J. Yang, K. Yu, and T. S. Huang, "Supervised translation-invariant sparse coding." in *CVPR'10*, 2010, pp. 3517–3524.

[104] S. van de Geer, *Empirical Processes in M-Estimation*, ser. Cambridge series in statistical and probabilistic mathematics. Cambridge, UK: Cambridge University Press, 2000.

[105] A. Banerjee, X. Guo, , and H. Wang, "On the optimality of conditional expectation as a Bregman predictor," *IEEE Transactions on Information Theory*, vol. 51, no. 7, pp. 2664ÂŰ–2669, July 2005.

[106] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Shift-invariant sparse representation of images using learned dictionaries," in *Machine Learning for Signal Processing, 2008. MLSP 2008. IEEE Workshop on*, 2008, pp. 145–150.

[107] J.M. Buhmann, "Empirical risk approximation: An induction principle for unsupervised learning," The University of Bonn, Tech. Rep. IAI-TR-98-3, 1998.

[108] A. Caponnetto and A. Rakhlin, "Stability properties of empirical risk minimization over Donsker classes," *Journal of Machine Learning Research*, vol. 7, pp. 2565–2583, 2006.

[109] A. Okabe, B. Boots, K. Sugihara, S. N. Chiu, and S. N. Chiu, *Spatial Tessellations : Concepts and Applications of Voronoi Diagrams*, 2nd ed. New York: Wiley, 2000.

[110] A. Gersho and R. Gray, *Vector Quantization and Signal Compression*. Boston: Kluwer Academic Publishers, 1992.

[111] H. Flanders, *Differential Forms with Applications to the Physical Sciences*. New York: Dover Publications, 1989.

[112] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. 39, pp. 1–38, 1977.

[113] A. Drémeau and C. Herzet, "An em-algorithm approach for the design of orthonormal bases adapted to sparse representations," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 2046–2049.

[114] S. S. Vempala, *The Random Projection Method*, ser. Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 2004.

[115] S. Dasgupta, "Learning mixtures of gaussians," in *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, 1999, pp. 634–644.

[116] R. Kannan and S. S. Vempala, *Spectral Algorithms*, ser. Foundations and Trends in Theoretical Computer Science. Now Publishers Inc, 2009.

[117] A. K. J. A. Vailya and H. Zhang, "Video clustering," Michigan State University, Technical Report, 1996.

[118] "Yuv video sequences," Available at http://trace.eas.asu.edu/yuv/index.html.

[119] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[120] A. Cichocki and S. Amari, *Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications*. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[121] M. Zibulevsky and B. A. Pearlmutter, "Blind source separation by sparse decomposition in a signal dictionary," *Neural Computation*, vol. 13, no. 4, pp. 863–882, April 2001.

[122] P. Bofill and M. Zibulevsky, "Underdetermined blind source separation using sparse representation," *Signal Processing*, vol. 81, no. 11, pp. 2353–2362, 2001.

[123] A. Belouchrani and M. G. Amin, "Blind source separation based on time-frequency signal representations," *IEEE Transactions on Signal Processing*, vol. 46, no. 11, pp. 2888–2897, 1998.

[124] L. Nguyen, A. Belouchrani, K. Abed-Meraim, and B. Boashash, "Separating more sources than sensors using time-frequency distributions," in *Proceedings of the International Symposium on Signal Processing and its Applications*, 2001, pp. 583–586.

[125] A. Jourjine, S. Rickard, and O. Yilmaz, "Blind separation of disjoint orthogonal signals: Demixing n sources from 2 mixtures," in *Proceedings of the IEEE ICASSP*, 2000, pp. 2985–2988.

[126] O. Yilmaz and S. Rickard, "Blind separation of speech mixtures via time-frequency masking," *IEEE Transactions on Signal Processing*, vol. 52, no. 7, pp. 1830–1847, 2004.

[127] F. Abrard and Y. Deville, "Blind separation of dependent sources using the time-frequency ratio of mixtures approach," in *Proceedings of the International Symposium on Signal Processing and its Applications*, 2003, pp. 1–4.

[128] ——, "A time-frequency blind signal separation method applicable to underdetermined mixtures of dependent sources," *Signal Processing*, vol. 85, no. 7, pp. 1389–1403, 2005.

[129] Y. Li, S. Amari, A. Cichocki, D. Ho, and S. Xie, "Underdetermined blind source separation based on sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 2, pp. 423–437, 2006.

[130] M. Xiao, S. Xie, and Y. Fu, "A novel approach for underdetermined blind sources separation in frequency domain," in *Proceedings of the International Symposium on Neural Networks*, 2005, pp. 484–489.

[131] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[132] C. Ding and T. Li, "Adaptive dimension reduction using discriminant analysis and K-means clustering," in *International Conference on Machine Learning*. ACM Press, 2007, pp. 84–405.

[133] F. D. L. Torre and T. Kanade, "Discriminative cluster analysis," in *International Conference on Machine Learning*. ACM Press, 2006, pp. 241–248.

[134] J. Ye, Z. Zhao, and H. Liu, "Adaptive distance metric learning for clustering," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007, pp. 1–7.

[135] J. Ye, Z. Zhao, and M. Wu, "Discriminative k-means for clustering," *Advances in Neural Information Processing Systems*, vol. 20, pp. 1649–1656, 2007.

[136] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural Computation*, vol. 12, pp. 2385–2404, October 2000.

[137] K. Fukunaga, *Introduction to Statistical Pattern Recognition (2nd ed.)*. San Diego, CA, USA: Academic Press Professional, Inc., 1990.

[138] G. Golub and C. Van Loan, *Matrix Computations*. Johns Hopkins University Press, 1996.

[139] M. Aharon and M. Elad, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[140] J.M. Duarte-Carvajalino *et.al.*, "Learning to sense sparse signals: simultaneous sensing matrix and sparsifying dictionary optimization," *IEEE Transactions on Image Processing*, vol. 18, no. 7, pp. 1395–1408, 2009.

[141] T. Poggio, R. Rifkin, S. Mukherjee, and P. Niyogi, "General conditions for predictivity in learning theory," *Nature*, vol. 428, no. 6981, pp. 419–422, 2004.

[142] A. Maurer and M. Pontil, "K-Dimensional Coding Schemes in Hilbert Spaces," *IEEE Transactions on Information Theory*, vol. 56, no. 11, pp. 5839–5846, 2010.

[143] D. Vainsencher and A. M. Bruckstein, "The Sample Complexity of Dictionary Learning," *Journal of Machine Learning Research*, vol. 12, pp. 3259–3281, 2011.

[144] S. Zhu *et.al.*, "Learning explicit and implicit visual manifolds by information projection," *Pattern Recognition Letters*, vol. 31, pp. 667ÂŰ–685, 2010.

[145] G. Yu, G. Sapiro, and S. Mallat, "Image modeling and enhancement via structured sparse model selection," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*. IEEE, 2010, pp. 1641–1644.

[146] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine learning*, vol. 40, no. 2, pp. 139–157, 2000.

[147] K. N. Ramamurthy, J. J. Thiagarajan, and A. Spanias, "Boosted dictionaries for image restoration based on sparse representations," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013.

[148] K. Ramamurthy, J. Thiagarjan, P. Sattigeri, and A. Spanias, "Ensemble sparse models for image analysis," *IEEE Transactions on Image Processing (Submitted)*, 2013.

[149] M. Elad, "Optimized projections for compressed-sensing," *IEEE Transactions on Signal Processing*, vol. 55, no. 12, pp. 5695–5702, 2007.

[150] "Uncompressed color image database," Available at http://vision.cs.aston.ac.uk/datasets/UCID/.

[151] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2. IEEE, 2005, pp. 1458–1465.

[152] Z. Jiang, Z. Lin, and L. Davis, "Learning a discriminative dictionary for sparse coding via label consistent K-SVD," in *IEEE CVPR*, 2011.

[153] K. Yu *et.al.*, "Nonlinear learning using local coordinate coding," *Proceedings of NIPS*, 2009.

[154] J. Wang *et.al.*, "Locality-constrained linear coding for image classification," in *Proceedings of IEEE CVPR*, 2010.

[155] R. Rigamonti, M. Brown, and V. Lepetit, "Are sparse representations really relevant for image classification?" in *IEEE CVPR*, 2011.

[156] D. Bradley and J. Bagnell, "Differential sparse coding," in *Proceedings of NIPS*, 2008.

[157] Q. Zhang and B. Li, "Discriminative K-SVD for dictionary learning in face recognition," in *IEEE CVPR*, 2010.

[158] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Sparse representations for pattern classification using learned dictionaries," in *Research and Development in Intelligent Systems XXV: Proceedings of AI-2008, The Twenty-eighth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, vol. 25. Springer, 2008, p. 33.

[159] S. Bengio, F. Pereira, Y. Singer, and D. Strelow, "Group sparse coding," in *Proceedings of NIPS*, 2009.

[160] S. Gao, I. Tsang, L. Chia, and P. Zhao, "Local features are not lonely âĂŞ laplacian sparse coding for image classification," in *IEEE CVPR*, 2010.

[161] K. Ramamurthy, J. Thiagarajan, P. Sattigeri, and A. Spanias, "Learning dictionaries with graph embedding constraints," in *Signals, Systems and Computers (ASILOMAR), 2012 Conference Record of the Forty Fifth Asilomar Conference on*. IEEE, 2012.

[162] M. Zheng, J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai, "Graph regularized sparse coding for image representation," *Image Processing, IEEE Transactions on*, vol. 20, no. 5, pp. 1327–1336, 2011.

[163] I. Ramirez, P. Sprechmann, and G. Sapiro, "Classification and clustering via dictionary learning with structured incoherence and shared features," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 3501–3508.

[164] Y. Lin, T. Liu, and C. Fuh, "Multiple kernel learning for dimensionality reduction," *IEEE Trans. on PAMI*, vol. 33, no. 6, pp. 1147–1160, Jun. 2011.

[165] S. Gou, Q. Li, and X. Zhang, "A new dictionary learning method for kernel matching pursuit," in *FSKD*, 2006, pp. 776–779.

[166] P. Vincent and Y. Bengio, "Kernel Matching Pursuit," *Machine Learning*, vol. 48, no. 1-3, pp. 165–187, 2002.

[167] S. Gao, I. Tsang, and L. Chia, "Kernel sparse representation for image classification and face recognition," *Computer Vision–ECCV 2010*, pp. 1–14, 2010.

[168] H. V. Nguyen *et. al.*, "Kernel dictionary learning," in *Proceedings of the IEEE ICASSP*, 2012.

[169] H. Zhang, A. C. Berg, M. Maire, and J. Malik, "Svm-knn: Discriminative nearest neighbor classification for visual category recognition," in *Proceedings of IEEE CVPR*, 2006, pp. 2126–2136.

[170] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal on Computer Vision*, vol. 60, pp. 91–110, November 2004.

[171] E. Shechtman and M. Irani, "Matching local self-similarities across images and ideos," in *Proceedings of IEEE CVPR*, June 2007.

[172] M. Pietikainen, A. Hadid, G. Zhao, and T. Ahonen, *Computer vision using Local Binary Patterns*, ser. Computational Imaging and Vision. Springer, 2011.

[173] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal on Computer Vision*, vol. 42, pp. 145–175, May 2001.

[174] A. Bosch, A. Zisserman, and X. Munoz, "Image classification using random forests and ferns," in *Proceedings of IEEE ICCV*, 2007.

[175] J. Mutch and D. G. Lowe, "Multiclass object recognition with sparse, localized features," in *Proceedings of IEEE CVPR*, vol. 1, 2006, pp. 11–18.

[176] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *Proceedings of IEEE CVPR*, 2005, pp. 994–1000.

[177] A. C. Berg and J. Malik, "Geometric blur for template matching," in *Proceedings of IEEE CVPR*, vol. 1, Los Alamitos, CA, USA, 2001.

[178] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[179] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Institute of Technology, Tech. Rep. 7694, 2007. [Online]. Available: http://authors.library.caltech.edu/7694

[180] P. Jain, B. Kulis, and K. Grauman, "Fast image search for learned metrics," in *Proceedings of IEEE CVPR*, Jun. 2008, pp. 1–8.

[181] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *Proceedings of IEEE CVPR*, Aug. 2008, pp. 1–8.

[182] D. Pham and S. Venkatesh, "Joint learning and dictionary construction for pattern recognition," in *Proceedings of IEEE CVPR*, vol. 0, 2008, pp. 1–8.

[183] J. C. Gemert, J. Geusebroek, C. Veenman, and A. W. Smeulders, "Kernel codebooks for scene categorization," in *Proceedings of ECCV*, 2008, pp. 696–709.

[184] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," in *IEEE CVPR*, June 2004.

[185] J. J. Corso *et. al.*, "Efficient multilevel brain tumor segmentation with integrated bayesian model classification," *Medical Imaging, IEEE Transactions on*, vol. 27, no. 5, pp. 629–640, 2008.

[186] M. Prastawa *et. al.*, "Automatic brain tumor segmentation by subject specific modification of atlas priors," *Academic Radiology*, vol. 10, no. 12, pp. 1341–1348, 2003.

[187] M.R. Kaus *et. al.*, "Automated segmentation of MR images of brain tumors," *Radiology*, vol. 218, no. 2, pp. 586–591, 2001.

[188] M. C. Clark and *et. al.*, "Automatic tumor segmentation using knowledge-based techniques," *Medical Imaging, IEEE Transactions on*, vol. 17, no. 2, pp. 187–201, 1998.

[189] T. Chan and L. Vese, "Active contours without edges," *Image Processing, IEEE Transactions on*, vol. 10, no. 2, pp. 266–277, 2001.

[190] S. Ho, E. Bullitt, and G. Gerig, "Level-set evolution with region competition: automatic 3-d segmentation of brain tumors," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 1. IEEE, 2002, pp. 532–535.

[191] N. Moon *et. al.*, "Model-based brain and tumor segmentation," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 1. IEEE, 2002, pp. 528–531.

[192] M. Prastawa *et. al.*, "A brain tumor segmentation framework based on outlier detection," *Medical Image Analysis*, vol. 8, no. 3, pp. 275–283, 2004.

[193] M.N. Ahmed *et. al.*, "A modified fuzzy c-means algorithm for bias field estimation and segmentation of MRI data," *Medical Imaging, IEEE Transactions on*, vol. 21, no. 3, pp. 193–199, 2002.

[194] C. H. Lee *et. al.*, "Segmenting brain tumors with conditional random fields and support vector machines," *Computer Vision for Biomedical Image Applications*, pp. 469–478, 2005.

[195] A. Hamamci *et. al.*, "Tumor-cut: Segmentation of brain tumors on contrast enhanced MR images for radiosurgery applications," *Medical Imaging, IEEE Transactions on*, no. 99, pp. 1–1, 2011.

[196] D. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[197] N. Cristianini and J. Shawe-Taylor, *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, 2000.

[198] H. V. Nguyen *et. al.*, "Kernel dictionary learning," in *Proceedings of the IEEE ICASSP*, 2012.

[199] J. Kwok and I. Tsang, "The pre-image problem in kernel methods," *Neural Networks, IEEE Transactions on*, vol. 15, no. 6, pp. 1517–1525, 2004.

[200] B. Brenner, D. Garcia, S. Anderson *et al.*, "Glomeruli and blood pressure. less of one, more the other?" *American journal of hypertension*, vol. 1, no. 4 Pt 1, p. 335, 1988.

[201] W. E. Hoy, J. F. Bertram, R. D. Denton, M. Zimanyi, T. Samuel, and M. D. Hughson, "Nephron number, glomerular volume, renal disease and hypertension," *Current opinion in nephrology and hypertension*, vol. 17, no. 3, pp. 258–265, 2008.

[202] J.-P. Bonvalet, M. Champion, F. Wanstok, G. Berjal *et al.*, "Compensatory renal hypertrophy in young rats: Increase in the number of nephrons," *Kidney Int*, vol. 1, no. 6, pp. 391–396, 1972.

[203] J. F. Bertram, M. C. Soosaipillai, S. D. Ricardo, and G. B. Ryan, "Total numbers of glomeruli and individual glomerular cell types in the normal rat kidney," *Cell and tissue research*, vol. 270, no. 1, pp. 37–45, 1992.

[204] S. C. Beeman, M. Zhang, L. Gubhaju, T. Wu, J. F. Bertram, D. H. Frakes, B. R. Cherry, and K. M. Bennett, "Measuring glomerular number and size in perfused kidneys using mri," *American Journal of Physiology-Renal Physiology*, vol. 300, no. 6, pp. F1454–F1457, 2011.

[205] Y. Chang, C. Hu, R. Feris, and M. Turk, "Manifold based analysis of facial expression," *Image and Vision Computing*, vol. 24, no. 6, pp. 605–614, 2006.

[206] H. S. Seung and D. D. Lee, "The manifold ways of perception," *Science*, vol. 290, no. 5500, pp. 2268–2269, 2000.

[207] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios, "Boostmap: A method for efficient approximate similarity rankings," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2. IEEE, 2004, pp. II–268.

[208] X. He, S. Yan, Y. Hu, and H.-J. Zhang, "Learning a locality preserving subspace for visual recognition," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 385–392.

[209] S. Roweis, L. K. Saul, G. E. Hinton *et al.*, "Global coordination of local linear models," *Advances in neural information processing systems*, vol. 2, pp. 889–896, 2002.

[210] D. Lowe and M. E. Tipping, "Neuroscale: Novel topographic feature extraction using rbf networks," *Advances in Neural Information Processing Systems*, pp. 543–549, 1997.

[211] M.-H. Yang, "Face recognition using extended isomap," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 2. IEEE, 2002, pp. II–117.

[212] H.-T. Chen, H.-W. Chang, and T.-L. Liu, "Local discriminant embedding and its variants," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 846–853.

[213] D. Cai, X. He, and J. Han, "Semi-supervised discriminant analysis," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007, pp. 1–7.

[214] L. K. Saul and S. T. Roweis, "An introduction to locally linear embedding," Tech. Rep., 2000.

[215] D. L. Donoho and C. Grimes, "Hessian eigenmaps: New locally linear embedding techniques for high-dimensional data," 2003.

[216] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in Neural Information Processing Systems 14*. MIT Press, 2001, pp. 585–591.

[217] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000.

[218] K. Weinberger and L. Saul, "Unsupervised learning of image manifolds by semidefinite programming," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, vol. 2, 2004, pp. 988–995 Vol.2.

[219] P. Niyogi, "Locality preserving projections," *Advances in neural information processing systems*, vol. 16, pp. 153–160, 2004.

[220] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 1, pp. 40–51, 2007.

[221] L. Cayton, "Algorithms for manifold learning," University of California, Technical report, 2005.

[222] Z. Lu and H. H. S. Ip, "Image categorization with spatial mismatch kernels," in *IEEE CVPR*, Jun. 2009, pp. 397 –404.

[223] L. Li and L. Fei-Fei, "What, where and who? classifying events by scene and object recognition," in *International Conference on Computer Vision*, Oct. 2007, pp. 1 –8.

[224] A. Rakotomamonjy, "Surveying and comparing simultaneous sparse approximation (or group lasso) algorithms," *Signal Processing*, vol. 91, no. 7, pp. 1505–1526, 2011.